Freeform Search

Database:
- US Pre-Grant Publication Full-Text Database
- **US Patents Full-Text Database**
- US OCR Full-Text Database
- EPO Abstracts Database
- JPO Abstracts Database
- Derwent World Patents Index
- IBM Technical Disclosure Bulletins

Term: 6363487.pn. or 5430845.pn. or 5822614.pn. or 5867730.pn. or 5870732.pn. or 5898843.pn. or 6044442.pn.

Display: 10 **Documents in Display Format:** TI **Starting with Number** 1

Generate: ○ **Hit List** ◉ **Hit Count** ○ **Side by Side** ○ **Image**

Search | Clear | Interrupt

Search History

DATE: Friday, April 30, 2004 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=YES; OP=OR			
L6	l4 same process$3	13	L6
L5	L4 and (plug adj5 play)	1	L5
L4	((logical or virtual) adj1 (ID or identification)) near5 (storage or memory)	35	L4
L3	L1 and (plug adj5 play)	1	L3
L2	L1 and (plug adj3 play)	1	L2
L1	(logical adj1 (ID or identification)) near5 (storage or memory)	22	L1

END OF SEARCH HISTORY

h e b b g e e e e e eg f ff e ch

Refine Search

Search Results -

Terms	Documents
((logical or virtual) adj1 (ID or identification)) near5 (storage or memory)	9

Database: US Pre-Grant Publication Full-Text Database; US Patents Full-Text Database; US OCR Full-Text Database; EPO Abstracts Database; JPO Abstracts Database; Derwent World Patents Index; IBM Technical Disclosure Bulletins

Search: L11

Refine Search | Recall Text | Clear | Interrupt

Search History

DATE: Friday, April 30, 2004 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR			
L11	((logical or virtual) adj1 (ID or identification)) near5 (storage or memory)	9	L11
DB=USPT; PLUR=YES; OP=OR			
L10	l7 and (plug adj5 play)	3	L10
L9	l4 and L7	0	L9
L8	l1 and L7	0	L8
L7	6363487.pn. or 5430845.pn. or 5822614.pn. or 5867730.pn. or 5870732.pn. or 5898843.pn. or 6044442.pn.	7	L7
L6	l4 same process$3	13	L6
L5	L4 and (plug adj5 play)	1	L5
L4	((logical or virtual) adj1 (ID or identification)) near5 (storage or memory)	35	L4
L3	L1 and (plug adj5 play)	1	L3
L2	L1 and (plug adj3 play)	1	L2
L1	(logical adj1 (ID or identification)) near5 (storage or memory)	22	L1

h e b b cg b e e ch

Refine Search

Search Results -

Terms	Documents
L12 and (plug adj5 play)	1

Database:
US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: L13

Refine Search

Recall Text

Clear

Interrupt

Search History

DATE: Friday, April 30, 2004 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=YES; OP=OR			
L13	L12 and (plug adj5 play)	1	L13
L12	((logical or virtual) adj1 (ID or identification)) same (storage or memory)	99	L12
DB=USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR			
L11	((logical or virtual) adj1 (ID or identification)) near5 (storage or memory)	9	L11
DB=USPT; PLUR=YES; OP=OR			
L10	l7 and (plug adj5 play)	3	L10
L9	l4 and L7	0	L9
L8	l1 and L7	0	L8
L7	6363487.pn. or 5430845.pn. or 5822614.pn. or 5867730.pn. or 5870732.pn. or 5898843.pn. or 6044442.pn.	7	L7
L6	l4 same process$3	13	L6
L5	L4 and (plug adj5 play)	1	L5
L4	((logical or virtual) adj1 (ID or identification)) near5 (storage or memory)	35	L4

h e b b cg b e e ch

L3	L1 and (plug adj5 play)	1	L3
L2	L1 and (plug adj3 play)	1	L2
L1	(logical adj1 (ID or identification)) near5 (storage or memory)	22	L1

END OF SEARCH HISTORY

Refine Search

Search Results -

Terms	Documents
L14 same (ID or identification or number)	13

Database: US Pre-Grant Publication Full-Text Database; US Patents Full-Text Database; US OCR Full-Text Database; EPO Abstracts Database; JPO Abstracts Database; Derwent World Patents Index; IBM Technical Disclosure Bulletins

Search: L15

Refine Search | Recall Text | Clear | Interrupt

Search History

DATE: Friday, April 30, 2004 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=YES; OP=OR			
L15	L14 same (ID or identification or number)	13	L15
L14	(plug adj5 play) same (storage or memory) same (logical or virtual)	36	L14
L13	L12 and (plug adj5 play)	1	L13
L12	((logical or virtual) adj1 (ID or identification)) same (storage or memory)	99	L12
DB=USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR			
L11	((logical or virtual) adj1 (ID or identification)) near5 (storage or memory)	9	L11
DB=USPT; PLUR=YES; OP=OR			
L10	l7 and (plug adj5 play)	3	L10
L9	l4 and L7	0	L9
L8	l1 and L7	0	L8
L7	6363487.pn. or 5430845.pn. or 5822614.pn. or 5867730.pn. or 5870732.pn. or 5898843.pn. or 6044442.pn.	7	L7
L6	l4 same process$3	13	L6

h e b b cg b e e ch

L5	L4 and (plug adj5 play)	1	L5
L4	((logical or virtual) adj1 (ID or identification)) near5 (storage or memory)	35	L4
L3	L1 and (plug adj5 play)	1	L3
L2	L1 and (plug adj3 play)	1	L2
L1	(logical adj1 (ID or identification)) near5 (storage or memory)	22	L1

END OF SEARCH HISTORY

Refine Search

Search Results -

Terms	Documents
4713734.pn. or 5129088.pn. or 5247647.pn. or 5524204.pn. or 5675761.pn. or 5701429.pn. or 5701462.pn. or 5842214.pn. or 5909691.pn. or 6081898.pn. or 6119131.pn.	11

Database:
US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: L16

Refine Search | Recall Text | Clear | Interrupt

Search History

DATE: Friday, April 30, 2004 Printable Copy Create Case

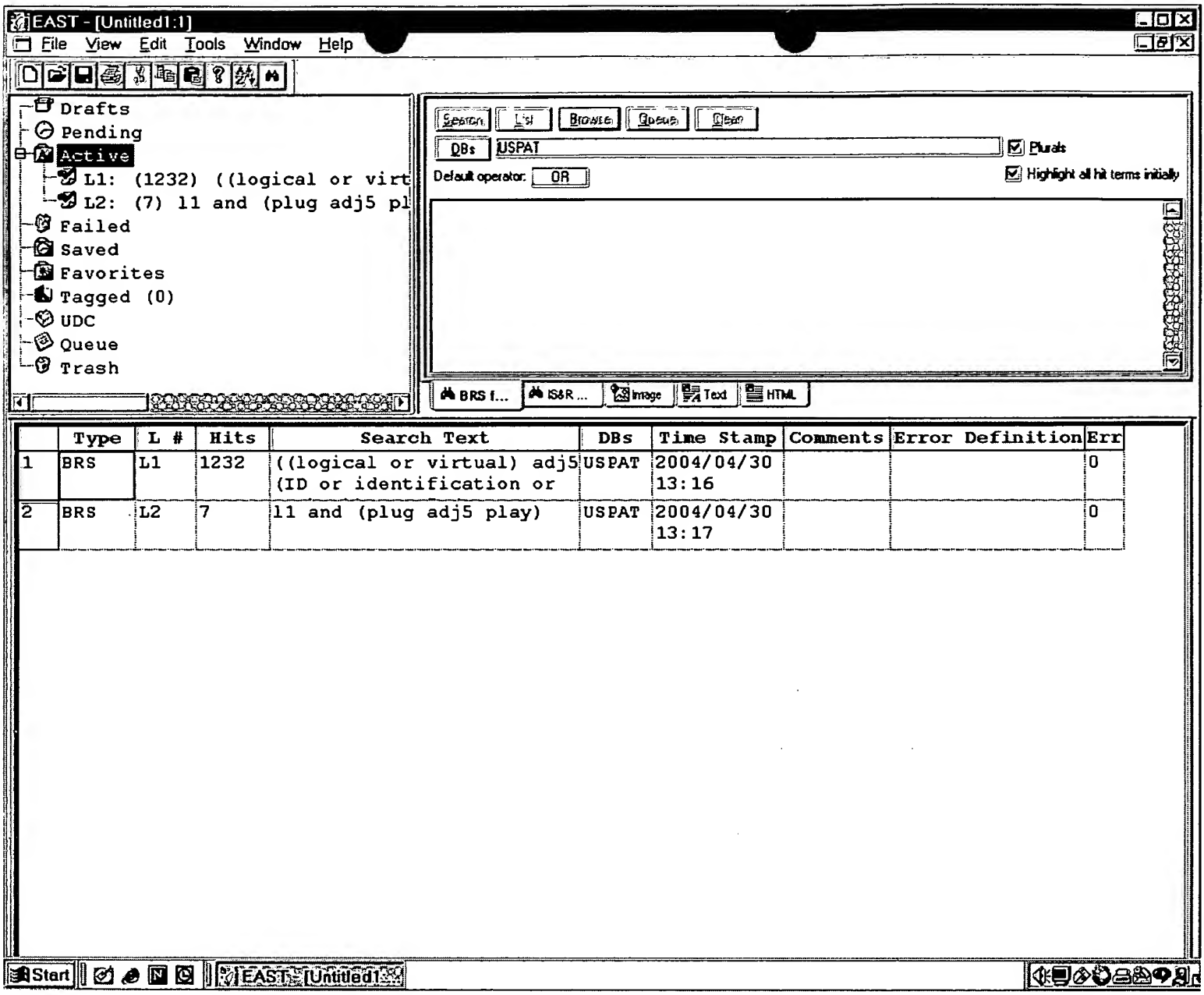
Set Name side by side	Query	Hit Count	Set Name result set
	DB=USPT; PLUR=YES; OP=OR		
L16	4713734.pn. or 5129088.pn. or 5247647.pn. or 5524204.pn. or 5675761.pn. or 5701429.pn. or 5701462.pn. or 5842214.pn. or 5909691.pn. or 6081898.pn. or 6119131.pn.	11	L16
L15	L14 same (ID or identification or number)	13	L15
L14	(plug adj5 play) same (storage or memory) same (logical or virtual)	36	L14
L13	L12 and (plug adj5 play)	1	L13
L12	((logical or virtual) adj1 (ID or identification)) same (storage or memory)	99	L12
	DB=USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR		
L11	((logical or virtual) adj1 (ID or identification)) near5 (storage or memory)	9	L11
	DB=USPT; PLUR=YES; OP=OR		
L10	l7 and (plug adj5 play)	3	L10

h e b b cg b e e ch

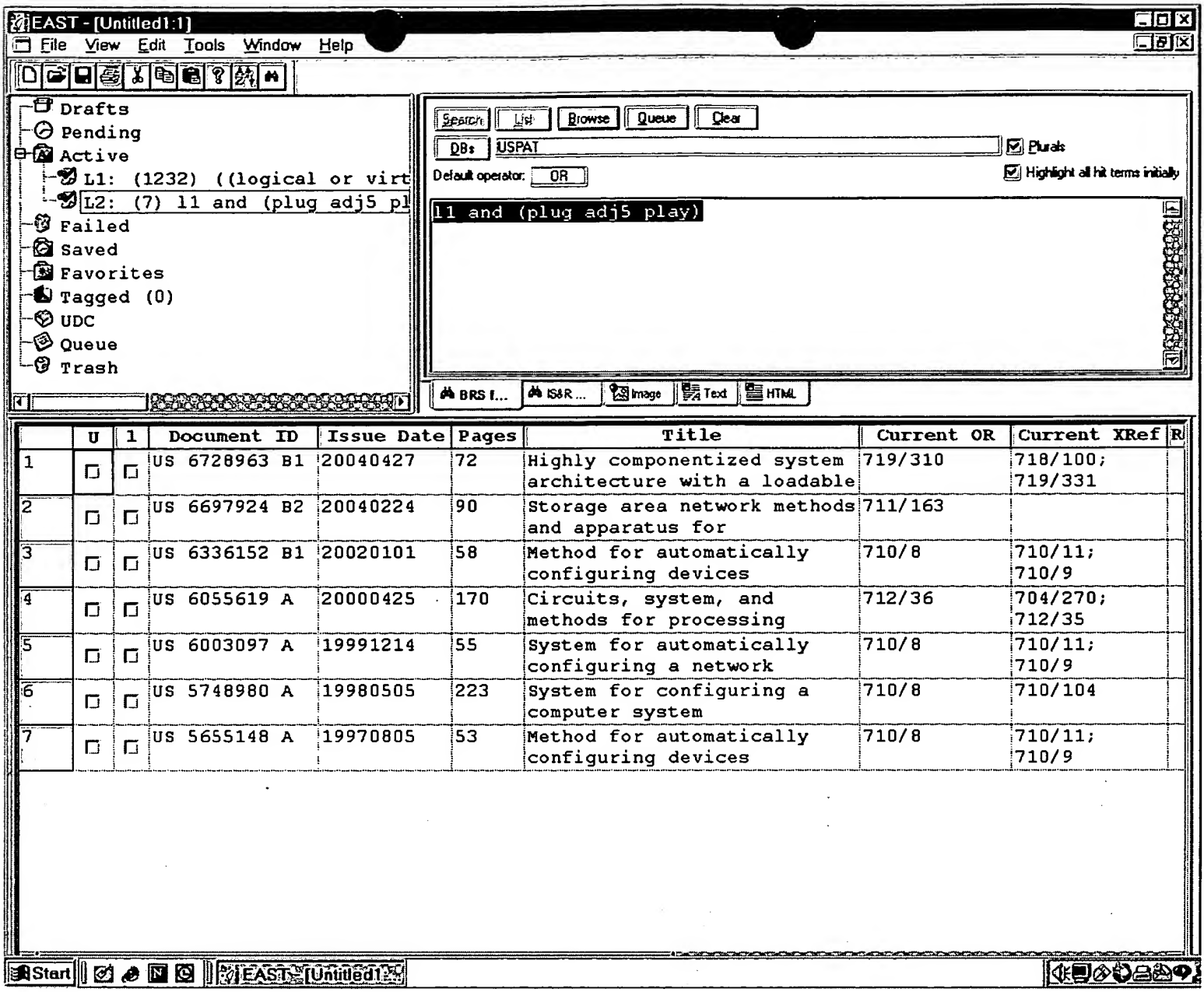
L9	l4 and L7	0	L9
L8	l1 and L7	0	L8
L7	6363487.pn. or 5430845.pn. or 5822614.pn. or 5867730.pn. or 5870732.pn. or 5898843.pn. or 6044442.pn.	7	L7
L6	l4 same process$3	13	L6
L5	L4 and (plug adj5 play)	1	L5
L4	((logical or virtual) adj1 (ID or identification)) near5 (storage or memory)	35	L4
L3	L1 and (plug adj5 play)	1	L3
L2	L1 and (plug adj3 play)	1	L2
L1	(logical adj1 (ID or identification)) near5 (storage or memory)	22	L1

END OF SEARCH HISTORY

h e b b cg b e e ch



	Type	L #	Hits	Search Text	DBs	Time Stamp	Comments	Error Definition	Err
1	BRS	L1	1232	((logical or virtual) adj5 (ID or identification or	USPAT	2004/04/30 13:16			0
2	BRS	L2	7	l1 and (plug adj5 play)	USPAT	2004/04/30 13:17			0



	U	1	Document ID	Issue Date	Pages	Title	Current OR	Current XRef	R
1	☐	☐	US 6728963 B1	20040427	72	Highly componentized system architecture with a loadable	719/310	718/100; 719/331	
2	☐	☐	US 6697924 B2	20040224	90	Storage area network methods and apparatus for	711/163		
3	☐	☐	US 6336152 B1	20020101	58	Method for automatically configuring devices	710/8	710/11; 710/9	
4	☐	☐	US 6055619 A	20000425	170	Circuits, system, and methods for processing	712/36	704/270; 712/35	
5	☐	☐	US 6003097 A	19991214	55	System for automatically configuring a network	710/8	710/11; 710/9	
6	☐	☐	US 5748980 A	19980505	223	System for configuring a computer system	710/8	710/104	
7	☐	☐	US 5655148 A	19970805	53	Method for automatically configuring devices	710/8	710/11; 710/9	

Hit List

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Search Results - Record(s) 1 through 10 of 13 returned.

☐ 1. Document ID: US 6643721 B1

L4: Entry 1 of 13 File: USPT Nov 4, 2003

US-PAT-NO: 6643721
DOCUMENT-IDENTIFIER: US 6643721 B1

TITLE: Input device-adaptive human-computer interface

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

☐ 2. Document ID: US 6591358 B2

L4: Entry 2 of 13 File: USPT Jul 8, 2003

US-PAT-NO: 6591358
DOCUMENT-IDENTIFIER: US 6591358 B2

TITLE: Computer system with operating system functions distributed among plural microcontrollers for managing device resources and CPU

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

☐ 3. Document ID: US 6460093 B1

L4: Entry 3 of 13 File: USPT Oct 1, 2002

US-PAT-NO: 6460093
DOCUMENT-IDENTIFIER: US 6460093 B1

TITLE: Automatic configuration of primary and secondary peripheral devices for a computer

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

☐ 4. Document ID: US 6427176 B1

L4: Entry 4 of 13 File: USPT Jul 30, 2002

US-PAT-NO: 6427176

h e b b g e e e f e ef b e

DOCUMENT-IDENTIFIER: US 6427176 B1
** See image for Certificate of Correction **

TITLE: Method and apparatus for maintaining system labeling based on stored configuration labeling information

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

---

☐ 5. Document ID: US 6349345 B1

L4: Entry 5 of 13 File: USPT Feb 19, 2002

US-PAT-NO: 6349345
DOCUMENT-IDENTIFIER: US 6349345 B1

TITLE: Autoconfigurable device that provisionally configures itself as the primary or secondary peripheral device depending on if another peripheral device is present

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

---

☐ 6. Document ID: US 6237091 B1

L4: Entry 6 of 13 File: USPT May 22, 2001

US-PAT-NO: 6237091
DOCUMENT-IDENTIFIER: US 6237091 B1
** See image for Certificate of Correction **

TITLE: Method of updating firmware without affecting initialization information

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

---

☐ 7. Document ID: US 6167463 A

L4: Entry 7 of 13 File: USPT Dec 26, 2000

US-PAT-NO: 6167463
DOCUMENT-IDENTIFIER: US 6167463 A

TITLE: Firm addressing for devices on a fibre channel arbitrated loop

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

---

☐ 8. Document ID: US 6145019 A

L4: Entry 8 of 13 File: USPT Nov 7, 2000

US-PAT-NO: 6145019
DOCUMENT-IDENTIFIER: US 6145019 A

h e b b g e e e f e ef b e

TITLE: Unconfigured device that automatically configures itself as the primary device if no other unconfigured device is present

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

---

☐ 9. Document ID: US 6044411 A

L4: Entry 9 of 13 File: USPT Mar 28, 2000

US-PAT-NO: 6044411
DOCUMENT-IDENTIFIER: US 6044411 A

TITLE: Method and apparatus for correlating computer system device physical location with logical address

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

---

☐ 10. Document ID: US 5881252 A

L4: Entry 10 of 13 File: USPT Mar 9, 1999

US-PAT-NO: 5881252
DOCUMENT-IDENTIFIER: US 5881252 A

TITLE: Method and apparatus for automatically configuring circuit cards in a computer system

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

---

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Terms	Documents
L1 and L3	13

Display Format: TI Change Format

Previous Page Next Page Go to Doc#

h e b b g e e e f e ef b e

First Hit Fwd Refs

Generate Collection Print

L4: Entry 12 of 13 File: USPT May 26, 1998

US-PAT-NO: 5758099
DOCUMENT-IDENTIFIER: US 5758099 A

TITLE: Plug and play protocol for bus adapter card

DATE-ISSUED: May 26, 1998

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Grieco; Frank Edward	Apex	NC		
Manson; Peter A.	Cary	NC		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
International Business Machines Corporation	Armonk	NY			02

APPL-NO: 08/ 657480 [PALM]
DATE FILED: May 29, 1996

INT-CL: [06] H01 J 13/00

US-CL-ISSUED: 395/282; 395/281, 395/285, 395/822
US-CL-CURRENT: 710/301; 710/105, 710/2

FIELD-OF-SEARCH: 395/281, 395/282, 395/283, 395/285, 395/309, 395/311, 395/822, 395/831, 395/834, 395/882, 395/883, 395/885, 395/892, 395/800, 395/527, 395/828, 395/830, 395/500, 395/651

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	5191653	March 1993	Banks et al.	395/821
☐	5239632	August 1993	Larner	395/311
☐	5420412	May 1995	Kowalski	235/492
☐	5517646	May 1996	Piccirillo et al.	395/700
☐	5535342	July 1996	Taylor	395/307
☐	5559965	September 1996	Oztaskin et al.	395/284

h e b b g e e e f c e e ge

☐	5590313	December 1996	Reynolds et al.	395/500
☐	5630174	May 1997	Stone, III et al.	395/883
☐	5634075	May 1997	Smith et al.	395/829
☐	5640594	June 1997	Gibson et al.	395/829
☐	5655148	August 1997	Richman et al.	395/828
☐	5689726	November 1997	Lin	395/830

ART-UNIT: 271

PRIMARY-EXAMINER: Harvey; Jack B.

ASSISTANT-EXAMINER: Phan; Raymond N.

ATTY-AGENT-FIRM: Phillips; Steven B. Galvin; Thomas F.

ABSTRACT:

A system and method of partitioning and providing communication to allow ISA Plug and Play protocol logic functions to be shared across multiple integrated circuits on a single Plug and Play compliant ISA bus adapter card as defined by the Plug and Play ISA Specification in a manner that minimizes the duplication of function.

11 Claims, 5 Drawing figures

L10: Entry 2 of 3 File: USPT Feb 2, 1999

DOCUMENT-IDENTIFIER: US 5867730 A
TITLE: Method for configuration of peripherals by interpreting response from peripherals to enable selection of driver file and altering configuration file to enable loading of selected driver file

Brief Summary Text (5):
Some computer interface boards, known as "plug and play" boards, have the ability to be identified and configured by the operating system. These boards contain extensive additional circuitry beyond that required to control the peripheral device. However, the identification of the boards is performed by the operating system. If the operating system is malfunctioning, the identification of boards cannot proceed.

Detailed Description Text (10):
If any of the software programs used to initialize the computer 10 are corrupted in any manner, the CPU 12 will be unable to execute instructions to properly configure the computer and start the DOS. For example, if the start-up program, such as IO.SYS, is corrupted, or the CONFIG.SYS data file is corrupted, the DOS will not function. Corruption of data files may be caused by the improper installation of computer hardware and its associated software, or the improper installation of an application software program. Corruption may also be caused by electrical transients, accidental erasure of data files, a computer virus, an actual hardware failure or the like. No matter what has caused the corruption of these critical data files, the computer 10 will not operate properly until the user corrects the problem. This is true even with plug and play boards, which are identified by the operating system, because the operating system cannot be started without the proper configuration of the computer 10. Similarly, operating systems such as Windows.RTM. 95 may have the ability to identify peripheral devices, such as the CD-ROM drive 30. However, the Windows 95 operating system cannot be started if there is an error in one of the required startup data files, such as CONFIG.SYS. Thus, the identification and configuration of interfaces cannot occur until the source of the error is determined and the corrupted file repaired or replaced. Unfortunately, the user often does not know the source of the problem. Furthermore, the typical user is unfamiliar with the computer hardware and software and is ill-prepared to load the proper drivers and to properly configure the computer 10.

Detailed Description Text (11):
The present invention is directed to a system and method that automatically identifies certain peripheral components and selects the correct software drivers for those identified components. Specifically, the present invention can identify the particular type of CD-ROM drive 30 installed in the computer 10. Following the identification, the present invention can select and load the appropriate software driver to properly configure the CD-ROM interface 30a to operate the CD-ROM drive. Similarly, the present invention can analyze the hard disk drive 28 and prepare the hard disk drive to receive data. The present invention is significantly different from other interface identification techniques, such as plug and play interface boards. Plug and play techniques are directed to the configuration of the hardware interface or controller for a peripheral device, rather than the software drivers that operate the peripheral device. For example, the CD-ROM drive interface 30a may be a plug and play type device. The plug and play techniques may be used to

identify and configure the CD-ROM drive interface 30a, with parameters such as I/O address, interrupt request line, direct memory access (DMA) request line, and the like. However, the CD-ROM drive 30 itself is not a plug and play device. The CD-ROM drive 30 requires the hardware control provided by the CD-ROM drive interface 30a and a software driver that operates the CD-ROM drive. The present invention is directed to a bootable system that identifies and loads the required software drivers.

First Hit Fwd Refs

Generate Collection Print

L10: Entry 2 of 3 File: USPT Feb 2, 1999

US-PAT-NO: 5867730
DOCUMENT-IDENTIFIER: US 5867730 A

TITLE: Method for configuration of peripherals by interpreting response from peripherals to enable selection of driver file and altering configuration file to enable loading of selected driver file

DATE-ISSUED: February 2, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Leyda; Jeff	Minneapolis	MN		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Micron Eletronics, Inc.	Nampa	ID			02

APPL-NO: 08/ 904208 [PALM]
DATE FILED: July 31, 1997

PARENT-CASE:
CROSS-REFERENCE TO RELATED APPLICATION This application is a division of U.S. patent application Ser. No. 08/634,006, filed Apr. 15, 1996 (now U.S. Pat. No. 5,794,032).

INT-CL: [06] G06 F 13/00, G06 F 9/445

US-CL-ISSUED: 395/830; 395/652, 395/200.5, 395/681, 395/500
US-CL-CURRENT: 710/10; 709/220, 713/2

FIELD-OF-SEARCH: 395/652, 395/500, 395/828, 395/830, 395/200.5, 395/200.52, 395/200.51, 395/681

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4589063	May 1986	Shah et al.	364/200
☐	5128995	July 1992	Arnold et al.	380/4
☐	5418918	May 1995	Vander Kamp et al.	395/375

h e b b g e e e f c e e eg e ge

☐	5463766	October 1995	Schieve et al.	395/650
☐	5581766	December 1996	Spurlock	395/652
☐	5598577	January 1997	Overfield	395/830
☐	5630076	May 1997	Saulpaugh et al.	395/284
☐	5655148	August 1997	Richman et al.	395/828
☐	5668992	September 1997	Hammer et al.	395/651
☐	5675831	October 1997	Caputo	395/830
☐	5701476	December 1997	Fenger	395/652
☐	5732282	March 1998	Provino et al.	395/830

ART-UNIT: 272

PRIMARY-EXAMINER: Lee; Thomas C.

ASSISTANT-EXAMINER: Perveen; Rehana

ATTY-AGENT-FIRM: Seed and Berry LLP

ABSTRACT:

A system for the automatic identification and configuration of a computer peripheral uses an initialization program to send one or more query instructions to a peripheral device such as a CD-ROM drive. In response to the query instructions, the CD-ROM drive replies with data that can be used to uniquely identify the model number or type of CD-ROM drive. The system then selects the appropriate software driver for the identified CD-ROM drive and loads the selected driver. The system can further verify proper operation of other peripheral devices, such as a hard disk drive, by using commands within the disk operating system. A floppy disk, or other computer readable storage media, contains the initialization program and a software driver file for every type of CD-ROM drive supported by the system. The system automatically identifies and loads the appropriate software driver without requiring the user to respond to technical questions related to the hardware.

16 Claims, 4 Drawing figures

h e b b g e e e f c e e eg e ge

First Hit Fwd Refs

Generate Collection Print

L10: Entry 1 of 3 File: USPT Apr 27, 1999

US-PAT-NO: 5898843
DOCUMENT-IDENTIFIER: US 5898843 A

TITLE: System and method for controlling device which is present in media console and system unit of a split computer system

DATE-ISSUED: April 27, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Crump, deceased; Dwayne T.	late of Lexington	KY		
Dawson, III; Marshall A.	Raleigh	NC		
Landry; John M.	Wake Forest	NC		
Mohre, II; Carl L.	Raleigh	NC		
Norris; Duane E.	Raleigh	NC		
Robinson; Eric F.	Raleigh	NC		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
International Business Machines Corporation	Armonk	NY			02

APPL-NO: 08/ 954996 [PALM]
DATE FILED: October 8, 1997

PARENT-CASE:
RELATED APPLICATIONS The present invention is believed to be related to the following pending applications: Application Ser. No.08/721,651, filed Sep. 23, 1996, and entitled "SPLIT SYSTEM PERSONAL COMPUTER" (further identified as Attorney Docket No. RP9-95-045); Application Ser. No. 08/721,653, filed Sep. 23, 1996, and entitled "MEDIA CONSOLE FOR A SPLIT SYSTEM PERSONAL COMPUTER" (further identified as Attorney Docket No. RP9-95-046); Application Ser. No. 08/721,650, filed Sep. 23, 1996, and entitled "MULTI-CONDUCTOR CABLE ARCHITECTURE AND INTERFACE FOR A SPLIT SYSTEM PERSONAL COMPUTER" (further identified as Attorney Docket No. RP9-96-009); Application Ser. No. 08/717,558, filed Sep. 23, 1996, and entitled "METHOD FOR INTERFACING A MEDIA CONSOLE AND A SYSTEM UNIT" (further identified as Attorney Docket No. RP9-96-014); and Application Ser. No. 08/946,407, filed Oct. 07, 1997, and entitled "LOCAL BUS IDE ARCHITECTURE FOR SPLIT COMPUTER SYSTEM" (further identified as Attorney Docket No. RP9-97-019).

INT-CL: [06] G06 F 13/00, G06 F 13/14

US-CL-ISSUED: 395/280; 395/823, 711/112, 360/39
US-CL-CURRENT: 710/100; 360/39, 711/112

FIELD-OF-SEARCH: 395/280, 395/281, 395/287, 395/823, 395/856, 711/112, 360/39, 360/113

h e b b cg b cc e

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected | Search ALL | Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	5016121	May 1991	Peddle et al.	360/39
☐	5050169	September 1991	Monett	371/21.2
☐	5113497	May 1992	Dewa	395/275
☐	5128995	July 1992	Arnold et al.	380/4
☐	5214695	May 1993	Arnold et al.	380/4
☐	5237690	August 1993	Bealkowski et al.	395/700
☐	5257378	October 1993	Sideserf et al.	395/700
☐	5299322	March 1994	Arai et al.	395/275
☐	5418960	May 1995	Munroe	395/700
☐	5428796	June 1995	Iskiyan et al.	395/728
☐	5437039	July 1995	Yuen	395/725
☐	5440740	August 1995	Chen et al.	395/650
☐	5442789	August 1995	Baker et al.	395/650
☐	5457785	October 1995	Kikinis et al.	395/308
☐	5497490	March 1996	Harada et al.	395/700
☐	5535415	July 1996	Kondou et al.	395/828
☐	5535419	July 1996	O'Brien	395/856
☐	5535420	July 1996	Kardach et al.	395/868
☐	5564060	October 1996	Mahalingaiah et al.	395/871
☐	5568649	October 1996	MacDonald et al.	395/868
☐	5678023	October 1997	Adams et al.	711/112

ART-UNIT: 271

PRIMARY-EXAMINER: Ray; Gopal C.

ATTY-AGENT-FIRM: Magistrale; Anthony N.

ABSTRACT:

Disclosed is a split computer system that includes a first housing coupled to a second housing with a multi-conductor cable. The first housing includes a first direct access storage device (DASD) having an opening for receiving a removable storage medium. The second housing is separate from the first housing and includes a central processing unit (CPU) coupled to a local bus and an expansion bus, a non-

h e b b cg b cc e

volatile storage device coupled to the local bus, a second DASD coupled to the local bus and a power supply. The system further includes first and second DASD controllers coupled to the first and second DASDs respectively. The cable has one end coupled to the first housing and another end coupled to the second housing for electrically connecting devices in the first housing to devices in the second housing. The second housing has a first interface coupled to the expansion bus and the cable. The first housing also includes a second interface coupled to the cable and the first DASD. The system is operative to detect an access to either the first or second DASD controller and disable a current DASD controller and enable the other DASD controller.

26 Claims, 13 Drawing figures

h e b b cg b cc e

First Hit Fwd Refs
End of Result Set

Generate Collection Print

L3: Entry 1 of 1 File: USPT Feb 24, 2004

DOCUMENT-IDENTIFIER: US 6697924 B2
TITLE: Storage area network methods and apparatus for identifying fiber channel devices in kernel mode

Brief Summary Text (98):
Further aspects of the invention provide an improved storage area network (SAN) of the type having a digital data processor, e.g., a host, in communication with one or more storage devices, e.g., a LUN and, further, of the type having a plug-and-play (PNP) manager that generates an event in response to a change in status of at least one of the storage devices.

Brief Summary Text (99):
The improvement is characterized, according to one aspect of the invention, by at least a selected process, that executes on the host (or other digital data processor), which references at least a selected one of the storage devices using a previously assigned logical identification, e.g., a LUN ID. The improvement is further characterized by the selected process responding to an event generated by the plug-and-play manager by querying for information the storage device (or an interface thereto) with respect to which the event was generated. From that information, the process generates a logical identification for the device.

Detailed Description Text (251):
According to the illustrated embodiment, the agent 40 prevents masked LUNs from appearing in the Device Manger of the Windows.TM. 2000 interface by setting a flag in the data structure normally sent by the plug-and-play manager (not illustrated) with the device state query. In addition, the illustrated embodiment prevents the plug-and-play manager from generating notifications to an operator of a host 12 from which a masked device has been removed. This is accomplished by setting a flag in the data structure normally sent by the plug-and-play manager along with the device capabilities query.

Detailed Description Text (253):
In normal operation of a Windows.TM. 2000 host, the plug-and-play manager (which is a conventional component of the Windows 2000 operating system) is initiated at boot-up and creates a data structure that it passes to the SCSI port driver 356. The port driver 356 populates that data structure with information regarding all found devices (e.g., SCSI addresses). The illustrated embodiment effects masking via the filter driver 354, which removes from that data structure information regarding fiber channel devices not listed in the table 354a. As a result, neither the plug-and-play manager nor the class driver become aware of masked devices and, hence, do not attempt to create disk objects for them.

Detailed Description Text (254):
To "add" back a LUN that was previously masked, the plug-and-play manager is initiated to create and send a new data structure to the port driver 356 to be filled in. The plug-and-play manager is initiated by issuing from "user mode" a call to the filter driver 354, which itself issues a kernel mode IO_INVALIDATE_DEVICE RELATIONS call. This causes the plug-and-play manager to issue

calls (IRPs) to the port driver 356, which causes refill of the data structure. Then the filter driver 354 again intercepts the response from the port driver 356, and removes any objects from the data structure that correspond to masked devices. Those skilled in art will appreciate that any other sequence of calls suitable for effecting refill of the data structure (e.g., DEVICE_RELATIONS) can be utilized.

Detailed Description Text (255):
To mask a LUN that is already available a command (i.e., REMOVE) is sent to the plug-and-play manager from "user mode" that identifies the device to be removed. The plug-and-play manager then removes all structures necessary for I/O (including disk objects). The filter driver 354 is active at all times to prevent any rescan from filling the data structure with a masked device.

Detailed Description Text (258):
To mask LUNs at the SCSI port level an upper filter driver 354 to the SCSI port driver 356 is used. The upper filter driver 356 catches Plug N Play request packets for devices on the SCSI port. The I/O request packet (IRP_MN_QUERY_DEVICE_RELATIONS) contains an array of all device objects attached to the SCSI port.

Detailed Description Text (261):
Once Windows 2000 is booted care must be taken when masking out LUNs to avoid a surprise remove. When an unmasked LUN needs to be masked a user mode uninstall must be done to unmount the partitions and remove the disk safely from the plug-and-play manager. The SCSI bus is then rescanned and the device driver removes the device object from the array after a user mode uninstall of the disk has been completed successfully.

Detailed Description Text (266):
By contrast, many functions within the host digital processors 12 inherently utilize physical device names or addresses to identify attached storage devices. For example, the plug-and-play manager within a Windows.TM. 2000 host identifies storage devices via physical device object names that include, among other things, port number, path number, target number and logical unit number.

Detailed Description Text (267):
The illustrated embodiment provides a mechanism for readily associating these physical device names/addresses with the corresponding LUN IDs, thereby, facilitating use of built-in host functions--e.g., plug-and-play manager detection services--to determine when the SAN storage devices have been added, removed, enabled, disabled, otherwise affected. Though the discussion here focuses on association of physical device object names of the type used by plug-and-play managers in the Windows.TM. 2000 environments, those skilled in the art will appreciate that the teachings are equally applicable to forming other such associations with this and other operating systems and operating system functions.

Detailed Description Text (268):
Referring to FIG. 36 by way of review, in normal operation of a Windows.TM. 2000 host, the plug-and-play manager (PNP) queries the SCSI port driver 356 for information regarding all devices known by it. The information includes data such as port number, path number, target number, and logical unit number for each found device. The PNP manager 386 generates from this a physical object for each device.

Detailed Description Text (286):
In the illustrated embodiment, common user mode code is utilized to use the common user mode interface 374 prior to an install of the filter device drivers 354 on the operating system, and immediately following a re-boot. The user mode code is only required once, because once the active topology is known, changes to that topology are not noticed until after a re-boot, especially on an operating system such as Windows NT and 2000. Although Windows 2000 adds the plug-and-play option, the

actual bus adapters cannot be hot plugged. Therefore, new or changed bus adapters are only recognized after re-boot.

h e b b g e e e f c e e eg e ge

First Hit Fwd Refs

End of Result Set

☐ Generate Collection Print

L3: Entry 1 of 1 File: USPT Feb 24, 2004

US-PAT-NO: 6697924
DOCUMENT-IDENTIFIER: US 6697924 B2

TITLE: Storage area network methods and apparatus for identifying fiber channel devices in kernel mode

DATE-ISSUED: February 24, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Swank; Raymond Matthew	Campbell	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
International Business Machines Corporation	Armonk	NY			02

APPL-NO: 09/ 972585 [PALM]
DATE FILED: October 5, 2001

INT-CL: [07] G06 F 12/14

US-CL-ISSUED: 711/163
US-CL-CURRENT: 711/163

FIELD-OF-SEARCH: 711/14, 711/112, 711/163

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	5363487	November 1994	Willman et al.	
☐	5430845	July 1995	Rimmer et al.	
☐	5822614	October 1998	Kenton et al.	
☐	5867730	February 1999	Leyda	
☐	5870732	February 1999	Fisher et al.	
☐	5898843	April 1999	Crump et al.	
	6044442	March 2000	Jesionowski	

h e b b g e e e f c e e eg e ge

☐				
☐	6115815	September 2000	Doragh et al.	
☐	6343324	January 2002	Hubis et al.	709/229

ART-UNIT: 2188

PRIMARY-EXAMINER: Ellis; Kevin L.

ATTY-AGENT-FIRM: Powsner; David J. Nutter, McClennen & Fish

ABSTRACT:

The invention provides an improved storage area network (SAN) of the type that has a host or other digital data processor whose ports are coupled to peripheral devices that include fiber channel or other SAN-class storage devices. Processes executing on the host (or other digital data processor) generate requests for access to those peripheral devices. A persistent store identifies ports coupled to SAN-class storage devices. This store can be loaded, for example, by a process that executes on the host in user mode. A filter executes on the host in kernel mode to block access to selected ones of those SAN-class storage devices.

20 Claims, 50 Drawing figures

L15: Entry 7 of 13 File: USPT Nov 28, 2000

DOCUMENT-IDENTIFIER: US 6154836 A
TITLE: Method and system for configuring plug and play devices for a computer operating system

Detailed Description Text (17):
Turning now to FIGS. 4A and 4B, there is illustrated a high level logic flow diagram that depicts steps for a process utilized to carry out the method and system for initialization of Plug and Play (PnP) and PCI devices in accordance with a preferred embodiment of the present invention. It can be appreciated by those skilled in the art that FIGS. 4A and 4B presents a self-consistent sequence of steps leading to a desired result. As depicted in blocks 100 and 102, the process in accordance with the present invention is started by initiating a Power On Self Test (POST). As depicted thereafter at block 104 an extended BIOS data area (EBDA) is initialized for non-boot devices within the computer system (i.e., computer system 20 of FIG. 1 and FIG. 2). Next, as depicted in block 108, the serial and parallel data area is copied from the BIOS data area (BDA) to the EBDA. The list of devices not necessary for booting is coded so that POST can step through it. It begins with a word (16 bits) signifying the start of PnP ISA devices. A specific value in the next word can signify the start of PCI devices as will be more fully described below. Otherwise as shown in block 110, it will contain and have assigned a Card Select Number (CSN) and logical device number of a PnP ISA device. During POST, ISA Plug and Play devices are classified based on the Memory Resource Descriptors. All ISA PnP devices without a ROM (such as audio and modem) are considered to be non-boot. This is because ISA PnP boot devices like video or SCSI adapters normally require their own BIOS ROM for initialization and run-time support.

First Hit Fwd Refs

Generate Collection Print

L15: Entry 7 of 13 File: USPT Nov 28, 2000

US-PAT-NO: 6154836
DOCUMENT-IDENTIFIER: US 6154836 A

TITLE: Method and system for configuring plug and play devices for a computer operating system

DATE-ISSUED: November 28, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Dawson, III; Marshall Allen	Longmont	CO		
Landry; John Matthew	Wake Forest	NC		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
International Business Machines Corporation	Armonk	NY			02

APPL-NO: 09/ 135119 [PALM]
DATE FILED: August 17, 1998

INT-CL: [07] G06 F 9/445

US-CL-ISSUED: 713/1
US-CL-CURRENT: 713/1

FIELD-OF-SEARCH: 713/1, 713/2, 713/100, 710/8-10

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	5634075	May 1997	Smith et al.	710/9
☐	5655148	August 1997	Richman et al.	710/8
☐	5748980	May 1998	Lipe et al.	710/8
☐	5822582	October 1998	Doragh et al.	713/2
☐	5999989	December 1999	Patel	710/1

h e b b g e e e f c e e eg e ge

ART-UNIT: 277

PRIMARY-EXAMINER: Heckler; Thomas M.

ATTY-AGENT-FIRM: Magistrale; Anthony N. Felsman, Bradley, Vaden, Gunter & Dillon, LLP

ABSTRACT:

A method and system are disclosed for configuring PnP devices for a computer operating system by initiating a power on self test (POST) within a computer system for configuring PnP and PCI devices. During the process of configuring PnP and PCI devices, a list is composed of devices that are not absolutely necessary for booting the system (e.g. modem or ethernet controller). While the PCI devices are configured, if the system has no usable IRQ's, POST takes one from a nonessential PnP ISA (Industry Standard Architecture) device in the list, and gives it to the PCI device. The POST operation searches for the presence of a PnP operating system option while progressing through the startup sequence (of bootable media), and activates or deactivates all devices, depending on the type of media being attempted. If the medium is the hard disk (where the PnP operating system option resides), all of the PnP devices in the list are deactivated. If the medium is any other type (where a PnP operating system option is not likely to reside) the devices are activated.

20 Claims, 6 Drawing figures

L15: Entry 12 of 13 File: USPT May 26, 1998

DOCUMENT-IDENTIFIER: US 5758099 A
TITLE: Plug and play protocol for bus adapter card

Detailed Description Text (26):
Additionally, main Plug and Play integrated circuit 202 maintains in its logic circuits or otherwise a "maximum logical device count" indicating the final (highest logical device number) device present or configured by main Plug and Play integrated circuit 202. One method for setting this logical device count is for the count to be stored in a locally accessible area of resource data memory device 203 coupled to integrated circuit 202 by connection 204. This method allows change in the count value on adapter card 102 by reprogramming resource data memory device 203 rather than the more expensive and difficult method of modifying logic circuits in integrated circuit 202.

Detailed Description Text (29):
Each slave Plug and Play integrated circuit 206 and 208 maintains in logic or otherwise a "maximum logical device count" indicating the final (highest logical device number) device present on the component, in the same manner as described above with respect to main Plug and Play integrated circuit 202. The "maximum logical device count" for slave integrated circuits 206 and 208 may be either hard wired within the integrated circuits 206 and 208, or presented to integrated circuits 206 and 208 via memory devices 220 and 221, respectively, or by programming via wiring input pins on each of integrated circuits 206 and 208. Slave Plug and Play integrated circuit 206, and successive slave integrated circuit(s) 208, have a logical device count higher than the main or proceeding slave integrated circuit.

Detailed Description Text (30):
Each slave Plug and Play integrated circuit 206 and 208 supports logic circuits for control and storage of the Plug and Play configuration register address, logical device number, read address, specific configuration registers for logical devices within this component, I/O range check logic (as appropriate) for those devices as described in the Plug and Play Specification.

First Hit Fwd Refs

Generate Collection Print

L15: Entry 12 of 13 File: USPT May 26, 1998

US-PAT-NO: 5758099
DOCUMENT-IDENTIFIER: US 5758099 A

TITLE: Plug and play protocol for bus adapter card

DATE-ISSUED: May 26, 1998

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Grieco; Frank Edward	Apex	NC		
Manson; Peter A.	Cary	NC		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
International Business Machines Corporation	Armonk	NY			02

APPL-NO: 08/ 657480 [PALM]
DATE FILED: May 29, 1996

INT-CL: [06] H01 J 13/00

US-CL-ISSUED: 395/282; 395/281, 395/285, 395/822
US-CL-CURRENT: 710/301; 710/105, 710/2

FIELD-OF-SEARCH: 395/281, 395/282, 395/283, 395/285, 395/309, 395/311, 395/822, 395/831, 395/834, 395/882, 395/883, 395/885, 395/892, 395/800, 395/527, 395/828, 395/830, 395/500, 395/651

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	5191653	March 1993	Banks et al.	395/821
☐	5239632	August 1993	Larner	395/311
☐	5420412	May 1995	Kowalski	235/492
☐	5517646	May 1996	Piccirillo et al.	395/700
☐	5535342	July 1996	Taylor	395/307
☐	5559965	September 1996	Oztaskin et al.	395/284

h e b b g e e e f c e e eg e ge

☐	5590313	December 1996	Reynolds et al.	395/500
☐	5630174	May 1997	Stone, III et al.	395/883
☐	5634075	May 1997	Smith et al.	395/829
☐	5640594	June 1997	Gibson et al.	395/829
☐	5655148	August 1997	Richman et al.	395/828
☐	5689726	November 1997	Lin	395/830

ART-UNIT: 271

PRIMARY-EXAMINER: Harvey; Jack B.

ASSISTANT-EXAMINER: Phan; Raymond N.

ATTY-AGENT-FIRM: Phillips; Steven B. Galvin; Thomas F.

ABSTRACT:

A system and method of partitioning and providing communication to allow ISA Plug and Play protocol logic functions to be shared across multiple integrated circuits on a single Plug and Play compliant ISA bus adapter card as defined by the Plug and Play ISA Specification in a manner that minimizes the duplication of function.

11 Claims, 5 Drawing figures

h e b b g e e e f c e e eg e ge

First Hit Fwd Refs

End of Result Set

Generate Collection Print

L15: Entry 13 of 13 File: USPT May 14, 1996

DOCUMENT-IDENTIFIER: US 5517646 A
TITLE: Expansion device configuration system having two configuration modes which uses automatic expansion configuration sequence during first mode and configures the device individually during second mode

Brief Summary Text (11):
Returning now to FIG. 1, control proceeds from step 106 to step 108, where the isolated expansion card is assigned a unique handle, which is also referred to as the card select number (CSN). The CSN is later used to select the expansion card. Expansion cards that have been assigned a non-zero CSN value will not participate in subsequent iterations of the isolation process in step 106. Once an expansion card is assigned a non-zero CSN value, it can respond to other bus commands. After the CSN is assigned for the expansion card, control proceeds from step 108 to 110, where the system BIOS performs resource data read cycles on the isolated expansion card. The resource data describes all the resource requirements of the Plug and Play ISA expansion card. The resource data includes such items as the Plug and Play version number, the number of logical devices (that is, the number of functions available on the Plug and Play expansion card), the logical device ID, compatible device ID, IRQ format, DMA format, I/O port descriptor, fixed location I/O port descriptor, memory range descriptor, identifier string and various other information. The resource data, along with the serial identifier described earlier in step 106, are conventionally stored in a serial EEPROM, which is typically 2K bits in size. The expansion card resource data is initially read into a resource data register located on the expansion card. After 8 bits have been loaded into the resource data register, a status flag on the expansion card is set indicating that the next byte of resource data is ready to be outputted. Thus, the system BIOS will read the resource data one byte at a time from the resource data register. This process is repeated until all the resource data has been read, in which case, control proceeds to step 112, where it is determined if all the Plug and Play expansion cards have been accessed. If not, control returns to step 104, where the configuration routine is reiterated. If all the cards have been accessed, then control proceeds to step 114.

Brief Summary Text (12):
In step 114, the system resources are assigned to each expansion card. For those expansion cards with more than one logical device, each logical device is assigned resources separately. Configuration registers are located on each expansion card for configuring the card's standard ISA resource usage for each logical device. To program the configuration registers, the system BIOS sends a command WAKE[CSN], along with write data to set the desired CSN. The selected expansion card enters into a CONFIG state, and all other expansion cards are forced into a SLEEP state. Next, a logical device number is written to the logical device number register to select the device that is to be programmed. After the proper logical device is selected, the configuration registers are written with the proper configuration values. Thus, memory configuration, I/O space configuration, interrupt request level configuration and DMA channel configuration are performed for each logical device. After the system resources for a logical device are assigned, the logical device is activated on the ISA bus. After all the configuration registers on an

expansion card are programmed, the expansion card is placed into the WAIT FOR KEY state. Thus, if it is desired at a later time to access the Plug and Play configuration registers, the initiation key can be issued by the system BIOS to access the desired expansion card. It is noted that the Plug and Play registers can be reprogrammed even in the operating system environment. This is desirable for docking stations, as well as for computer systems that support hot insertion capability and power management. After all the expansion cards have been configured and all the logical devices have been activated, the system BIOS exits the Plug and Play configuration routine, and the standard POST procedure is executed.

First Hit Fwd Refs

End of Result Set

Generate Collection Print

L15: Entry 13 of 13 File: USPT May 14, 1996

US-PAT-NO: 5517646
DOCUMENT-IDENTIFIER: US 5517646 A

TITLE: Expansion device configuration system having two configuration modes which uses automatic expansion configuration sequence during first mode and configures the device individually during second mode

DATE-ISSUED: May 14, 1996

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Piccirillo; Gary J.	Houston	TX		
Welker; Mark W.	Spring	TX		
Thayer; John S.	Houston	TX		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Compaq Computer Corp.	Houston	TX			02

APPL-NO: 08/ 233032 [PALM]
DATE FILED: April 25, 1994

INT-CL: [06] G06 F 13/00, G06 F 9/00

US-CL-ISSUED: 395/700; 395/500, 364/929.4, 364/929.5, 364/975.2, 364/945, 364/929.2, 364/DIG.2
US-CL-CURRENT: 713/1; 710/10

FIELD-OF-SEARCH: 395/700, 395/725, 395/775, 395/800, 395/575, 395/500, 395/325, 395/200, 395/200.01

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4373181	February 1983	Chrisholm et al.	395/400
☐	4578773	March 1986	Desai et al.	395/275
☐	4750136	June 1988	Arpin et al.	364/514
☐	4760553	July 1988	Buckley et al.	364/900

h e b b g e e e f c e e eg e ge

☐	4800302	January 1989	Marum	326/10
☐	5038320	August 1991	Heath et al.	364/900
☐	5111423	May 1992	Kopec, Jr. et al.	395/500
☐	5161102	November 1992	Griffin et al.	395/800
☐	5247682	September 1993	Kondou et al.	395/700
☐	5257387	October 1993	Richek et al.	395/800
☐	5317750	May 1994	Wickersheim et al.	395/725
☐	5329634	July 1994	Thompson	395/500
☐	5367640	November 1994	Hamilton et al.	395/827
☐	5379431	January 1995	Lemon et al.	395/700
☐	5418960	May 1995	Munroe	395/700

OTHER PUBLICATIONS

Jonathan Cassell; "Electronic Buyers News" Nov. 15, 1993, p. 60.
Plug and Play ISA Specification, Version 1.02, pp. 4-25, 27-28, 51-53, 60-64, (Mar. 15, 1994).
Plug and Play BIOS Specification, Version 1.0A, pp. 4-27, 38-46, (Mar. 10, 1994).

ART-UNIT: 232

PRIMARY-EXAMINER: An; Meng-Ai

ATTY-AGENT-FIRM: Pravel, Hewitt, Kimball & Krieger

ABSTRACT:

A circuit for configuring a Plug and Play expansion card in one of three ways. The first is the standard Plug and Play configuration method, wherein expansion cards go through the isolation process to obtain unique Card Select Numbers (CSN). This method requires the existence of a dedicated serial EEPROM to store the system resource data for the expansion cards. However, when an expansion card is directly mounted onto a system board, it becomes a system board device. This allows the separate serial EEPROM to be removed. To implement, two alternative configuration modes are provided, wherein the expansion card can be configured under main CPU control. In these alternative modes, the configuration data is stored in the main system BIOS ROM. In the first mode, a register in the expansion card is mapped to a fixed ISA I/O address. In the second mode, the register is controlled by a dedicated pin, thus allowing it to be mapped to any ISA I/O address. To determine which configuration mode is used by the expansion card, pullup or pulldown resistors are connected to certain expansion card output pins. A second embodiment is also described wherein a static random access memory (SRAM) is utilized to store the serial identifier and the resource data. In this embodiment, the system BIOS initially writes the serial identifier and resource data into the SRAM. After this is done, a Plug and Play configuration process is invoked, in which the serial identifier is retrieved from the SRAM rather than the serial EEPROM.

22 Claims, 13 Drawing figures

First Hit Fwd Refs

End of Result Set

Generate Collection Print

L17: Entry 1 of 1 File: USPT Sep 12, 2000

DOCUMENT-IDENTIFIER: US 6119131 A
** See image for Certificate of Correction **
TITLE: Persistent volume mount points

Detailed Description Text (56):
In this section of the detailed description, a particular implementation of the invention is described that executes as part of the Microsoft Windows NT 5.0 operating system kernel. In the implementation illustrated in FIG. 4, the mount manager 401 and four other kernel modules work together to provide a user with access to data stored on a physical storage device 411 (shown as a fixed hard disk): a plug and play manager 403, an object manager 405, a partition manager 407, and at least one volume manager 409.

Detailed Description Text (57):
The mount manager 401 is not limited to use with only devices that adhere to the partition manager and volume manager architectures described below. The mount manager 401 will manage any device which registers with the plug and play manager 403 which has some mechanism for reporting a device name and a unique identifier that is persistent between boots. The partition manager 407 and the volume manager 409 are shown and described for the sake of clarity in understanding the invention.

Detailed Description Text (62):
When the partition manager 407 is initialized, it requests notification from the plug and play manager of all volume managers 409 registered in the system. As each volume manager 409 registers, the plug and play system notifies the partition manager 407 which maintains a list of the volume managers 409 ordered by their arrival in the system.

Detailed Description Text (63):
When the physical device 411 is detected by the plug and play manager 403 upon booting the system, the plug and play manager 403 determines the formatted characteristics of the physical device 411. The plug and play manager 403 loads the appropriate device driver to handle I/O access to the device. The device driver enumerates the partition device objects 421-424 used to access the data. As each partition device object 422-424 not representative of the entire device is enumerated by the device driver, the partition manager 407 "captures" the partition device object 422-424 before the driver registers the object with the plug and play manager 403. The partition manager 407 presents each partition device object 422-424 to the volume managers 409 in the order in which the volume managers 409 arrived in the system. Because each partition device object 424-424 is associated with at least one logical volume, the volume manager 409 responsible for the corresponding logical volume(s) accepts the device object.

Detailed Description Text (64):
When a volume manager 409 has received a sufficient number of partition device objects corresponding to a particular logical volume, the volume manager 409 assigns a device name to the logical volume and enumerates a volume device object

431, 432, 433 or 434 for the logical volume containing the device name and the unique volume identifier for the logical volume. In the NT 5.0 embodiment, the device name is guaranteed to be unique during a boot session, while the unique volume identifier is guaranteed to be unique across boot sessions. A counted string is used as the unique volume identifier in the NT 5.0 environment but a fixed length string can be equally applicable in other operating system environments. The counted string is as long as necessary to uniquely identify the device in the computer across multiple boot sessions. The volume device object 431-434 is stored by the object manager 405 in the device hierarchy by its device name. The volume manager 409 informs the plug and play manager 403 of the creation of the volume device object 431-434.

Detailed Description Text (65):
Each volume device object 431-434 is presented to the mount manager 401 by the plug and play manager 403. The mount manager 401 queries the volume device object 431-434 for its device name and unique volume identifier. Because of the indeterminate length of the unique volume identifier, the volume device object returns a byte count along with the string.

Detailed Description Text (68):
In order to facilitate the assignment of drive letter redirected names to logical volumes, the mount manager data structure(s) 441 contains an entry for each of the twenty-four drive letters assignable to fixed hard disks, i.e., .backslash.DosDevices.backslash.C:.backslash.-.backslash.DosDevices.backslash.Z:.backslash.. The entries are sorted in alphabetical order. Upon the initial boot of the computer, only the logical boot volume is assigned a drive letter (such as .backslash.DosDevices.backslash.C:.backslash.) When a drive letter is requested for a logical volume by the plug and play manager 403 during the initial boot process, the mount manager 401 assigns the next available drive letter by storing the unique volume identifier in the corresponding entry in the data structure(s) 441. The mount manager 401 requests that the object manager 405 create a symbolic link object representing the association between the drive letter and the volume device name.

Detailed Description Text (69):
On subsequent boots, each logical device is assigned its previous drive letter if one is present in the data structure(s) 441. If a new logical device is introduced into the system during the boot process, the plug and play manager 403 must request the assignment of a drive letter. On the other hand, a new logical volume that is introduced during a boot session is automatically assigned the next available drive letter.

Detailed Description Text (70):
The plug and play manager 403 also informs the mount manager 401 when a logical volume will be temporarily removed from the boot session. The mount manager 401 deletes the device names, if present, from the appropriate entries in the data structure(s) 441. The mount manager 401 also causes the object manager 405 to retire the symbolic link objects between the volume device name and the drive letter and/or persistent mount name.

Detailed Description Text (81):
where the pointer to a device object is passed to the mount manager 401 by the plug and play manager 403.

Detailed Description Text (86):
The mount manager 401 also provides an API with the plug and play manager 403 for managing the association between unique volume identifiers and the redirected names:

h e b b g e e e f c e e eg e ge

Detailed Description Text (95):
QueryPoints is called by the plug and play manager 403 to retrieve the entry in the mount manager data structure 441 for a logical volume. The plug and play manager 403 specifies the drive letter, the unique volume

Detailed Description Text (99):
In order to preserve the historical drive letter assignments across boot sessions, the mount manager 401 does not automatically assign a drive letter to a logical volume during the boot process unless the logical volume had previously been assigned a drive letter. Therefore, the plug and play manager 403 uses NextDriveLetter to request that the mount manager 401 assign a drive letter to the logical volume associated with the device name specified in the call. NextDriveLetter returns the current drive letter and an indication of whether a drive letter was assigned. A drive letter cannot be assigned if no drive letter is available or if the logical volume represented by the device name is already assigned a drive letter. The plug and play manager 403 can also use AutoDriveLetter once the historical assignments have been made to request the mount manager 401 assign drive letters to all subsequent logical volumes upon arrival.

CLAIMS:

11. A computer-readable medium having computer-executable components comprising:

a plug and play manager for detecting the presence of a physical device in a computer system and for assigning a device driver responsibility for controlling access to the physical device;

a partition manager communicatively coupled to the device driver for capturing partition device objects enumerated from the physical device by the device driver, wherein each partition device object corresponds to a portion of the physical device, the partition manager further communicatively coupled to the plug and play manager;

a volume manager communicatively coupled to the partition manager for creating a volume device object from at least one partition device object captured by the partition manager, for assigning a device name to the logical volume represented by the volume device object, and further communicatively coupled to the plug and play manager for registering the creation of the volume device object, wherein the volume device object comprises the device name and a unique volume identifier for the logical volume;

a mount manager communicatively coupled to the plug and play manager for receiving notification of the creation of the volume device object, for establishing a persistent association between the unique volume identifier of the volume device object and a unique mount name, for establishing a persistent association between the unique volume identifier and a drive letter when requested by the plug and play manager, and for establishing a persistent association on a host logical volume between a volume mount point on the host logical volume and a unique mount name for a logical volume object for a target logical volume mounted at the volume mount

point; and

an object manager communicatively coupled to the partition manager, the volume manager, and the mount manager for managing the partition device objects and the volume device object, for creating a symbolic link object for the mount name that causes a reference to the mount name to be redirected to the volume device object, for creating a symbolic link object for the drive letter that causes a reference to the drive letter to be redirected to the volume device object, and for creating a symbolic link object for the mount name that causes a reference to the mount name to be redirected to the volume device object for the target logical volume.

First Hit Fwd Refs

End of Result Set

☐ Generate Collection Print

L17: Entry 1 of 1 File: USPT Sep 12, 2000

US-PAT-NO: 6119131
DOCUMENT-IDENTIFIER: US 6119131 A
**** See image for Certificate of Correction ****

TITLE: Persistent volume mount points

DATE-ISSUED: September 12, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Cabrera; Luis Felipe	Bellevue	WA		
Kusters; Norbert P.	Woodinville	WA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Microsoft Corporation	Redmond	WA			02

APPL-NO: 09/ 097061 [PALM]
DATE FILED: June 12, 1998

PARENT-CASE:
RELATED APPLICATIONS This application is related to the co-assigned and co-filed and pending U.S. patent application Ser. No. 09/096,772 titled "Logical Volume Mount Manager" and pending U.S. patent application Ser. No. 09/096,540 "Persistent Names for Logical Volumes," which are hereby incorporated by reference.

INT-CL: [07] C06 F 12/00

US-CL-ISSUED: 707/203; 707/200, 707/201, 707/202, 707/204, 707/205, 707/206
US-CL-CURRENT: 707/203; 707/200, 707/201, 707/202, 707/204, 707/205, 707/206

FIELD-OF-SEARCH: 707/1, 707/4, 707/101, 707/103, 707/203, 707/200, 707/201, 707/202, 707/204, 707/205, 707/206, 711/4, 711/118, 714/15, 714/20, 709/206, 395/185.05, 710/260

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	5032979	July 1991	Hecht et al.	713/201

h e b b g e e e f c e e eg e ge

☐	5465365	November 1995	Winterbottom	707/101
☐	5623666	April 1997	Pike et al.	700/200
☐	5671414	September 1997	Nicolet	709/328
☐	5689706	November 1997	Rao et al.	707/201
☐	5724512	March 1998	Winterbottom	709/226
☐	5870734	February 1999	Kao	707/2
☐	5931935	August 1999	Cabrera et al.	710/260
☐	5991777	November 1999	Momoh et al.	707/205

ART-UNIT: 271

PRIMARY-EXAMINER: Black; Thomas G.

ASSISTANT-EXAMINER: Mizrahi; Diane D.

ATTY-AGENT-FIRM: Schwegman, Lundberg, Woessner & Kluth, P.A.

ABSTRACT:

Information regarding volume mount points hosted by a logical volume are stored on the physical device underlying the logical volume so that the relationships between the host logical volume and target logical volumes mounted on the volume mount points can be reconstituted when the system containing the logical volumes is rebooted, when the underlying physical devices are moved with the system, and when the logical volumes are transported to a different system. A data structure stored on the physical device contains the directory name of the volume mount point and the unique identifier and a globally unique mount name of the target logical volume mounted at the volume mount point. When the target logical volume is present in the system, symbolic links are created to relate the volume mount point name to a device name for the target logical volume so that pathnames containing the directory junction name are resolved correctly. If the target volume is not present in the system, the corresponding symbolic link does not exist, so an incorrect logical volume cannot be mounted onto a volume mount point. Furthermore, because the logical volumes contain the directory junction information, the namespace representing the logical volumes is self-describing so that neither user knowledge nor intervention is required to reconstitute the namespace.

13 Claims, 9 Drawing figures

h e b b g e e e f c e e eg e ge

US006697924B2

(12) **United States Patent**
Swank

(10) Patent No.: **US 6,697,924 B2**
(45) Date of Patent: **Feb. 24, 2004**

(54) **STORAGE AREA NETWORK METHODS AND APPARATUS FOR IDENTIFYING FIBER CHANNEL DEVICES IN KERNEL MODE**

(75) Inventor: Raymond Matthew Swank, Campbell, CA (US)

(73) Assignee: International Business Machines Corporation, Armonk, NY (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 102 days.

(21) Appl. No.: 09/972,585

(22) Filed: Oct. 5, 2001

(65) Prior Publication Data

US 2003/0172239 A1 Sep. 11, 2003

(51) Int. Cl.[7] G06F 12/14
(52) U.S. Cl. 711/163
(58) Field of Search 711/14, 112, 163

(56) References Cited

U.S. PATENT DOCUMENTS

5,353,487 A 11/1994 Wilman et al.
5,530,845 A 7/1996 Hamner et al.
5,822,514 A 10/1998 Kerrow et al.
5,867,730 A 2/1999 Leyda
5,890,732 A 2/1999 Fisher et al.
5,958,043 A 4/1999 Camp et al.
6,044,442 A 3/2000 Jankowski
6,115,815 A 9/2000 Keogh et al.
6,343,324 B1 * 1/2002 Hubis et al. 709/229

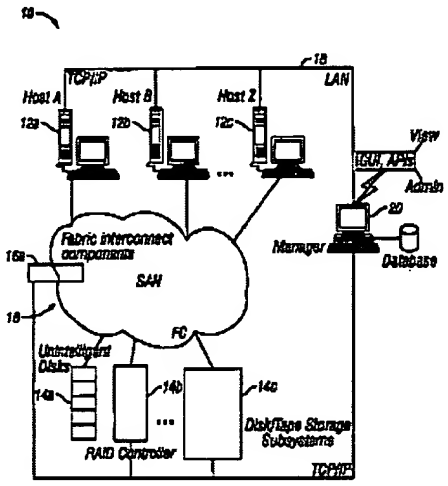
* cited by examiner

Primary Examiner—Kevin L. Ellis
(74) *Attorney, Agent, or Firm*—David J. Powsner; Nutter, McClennen & Fish

(57) **ABSTRACT**

The invention provides an improved storage area network (SAN) of the type that has a host or other digital data processor whose ports are coupled to peripheral devices that include fiber channel or other SAN-class storage devices. Processes executing on the host (or other digital data processor) generate requests for access to those peripheral devices. A persistent store identifies ports coupled to SAN-class storage devices. This store can be loaded, for example, by a process that executes on the host in user mode. A filter executes on the host in kernel mode to block access to selected ones of those SAN-class storage devices.

20 Claims, 45 Drawing Sheets



L3: Entry 4 of 20 File: USPT Sep 30, 2003

DOCUMENT-IDENTIFIER: US 6628999 B1
TITLE: Single-chip audio system volume control circuitry and methods

Detailed Description Text (81):
Codec 100 operates in conjunction with a number of other associated Plug & Play devices 909 also coupled to the ISA bus of independent blocks 909 that are mapped to the ISA Bus. Each block 909 has associated with it a set of resource requirements and associated configuration registers, organized into groups called physical devices. TABLE 6 below lists the maximum resource requirements for each physical device. The Intel/Microsoft Plug & Play specification organizes devices into logical groupings (logical devices) comprised of one or more physical devices.

Detailed Description Text (129):
As indicated above, Plug-n-Play organizes physical devices into groups of logical devices. A logical device may be comprised of up to four non-contiguous Memory Address ranges, eight non-contiguous I/O Address ranges, two Interrupts, and two DMA channels. Codec 100 only supports I/O, interrupts, and DMA.

Detailed Description Text (130):
Codec 100 has a fixed physical-to-logical device mapping summarized in TABLE 21. The Plug-n-Play resource data must match the Logical-to-Physical device mapping defined in TABLE 20. Controller 103 firmware translates Plug-n-Play logical device configuration cycles into writes of the appropriate hardware configuration registers.

Detailed Description Text (166):
The instructions and commands in the foregoing exemplary programmed sequence can be described as follows; Send Crystal Key--The "Crystal Key" is not a command but a sequence of 32 bytes that are written in succession. When Codec 100 receives the correct sequence of 32 bytes the Plug-n-Play logic of Codec 100 transitions to the Configuration State. The configuration registers of Codec 100 may only be modified when Codec 100 is in the Configuration State. Program the CSN (Card Select Number) 0x6--The CSN number for Codec 100 may optionally be programmed by executing this command. This command is executed by writing a 0x6 followed by the 8-bit CSN number. If this command is not used then the CSN number for Codec 100 will default to zero. Select Logical Device (0x15)--The configuration registers of codec 100 are programmed one logical device at a time. This command is executed by writing a 0x15 followed by an 8-bit logical device number. Codec 100 supports eight physical devices (0:7) as previously noted. IO Port Base Address 0 (0x47)--This command is executed by writing a 0x47 followed by a write of the low byte of the I/O base address, and a write 6f the high byte of the I/O base address. IO Port Base Address I (0x48)--This command is executed by writing a 0x48 followed by a write of the low byte of the I/O bang address, and A write of the high byte of the I/O base address. IO Port Base Address 2 (0x42)--This command is executed by writing a 0x42 followed by a write of the low byte of the I/O base address, and a write of the high byte of the I/O base address. Interrupt Select 0 (0x2A)--This command is executed by writing a 0x22 followed by a write of the interrupt line to generate an interrupt on. Interrupt Select 1 (0x27)--This command is executed by writing a 0x27 followed by a write of the interrupt line to generate an interrupt on. DMA Select 0 (WA)--

This command is executed by writing a 0x2A followed by a write of the DMA channel that is to be used. DMA Select 1 (0x25)--This command is executed by writing a 0x25 followed by a write of the DMA channel that is to be used. Activate Logical Device (0x33)--This command is executed by writing a 0x33 followed by a byte of one to activate the currently selected logical device. Deactivate Logical Device (0x33)--This command is executed by writing a 0x33 followed by a byte of zero to deactivate the currently selected logical device. Activate Codec 100 (6x79) The configuration data are processed and transferred to the appropriate Codec 100 registers upon execution of this command. This command puts Codec 100 into the Wait_For_Key_State.

Detailed Description Text (167):
Once a Plug & Play sequence has transpired each logical device, including Codec 100, will have an I/O base address assigned to it. This assigned base address is stored in each I/O base address register. ISA bus address bits A12 . . . A0 are compared with the values stored in the I/O base address registers, and if a match is found, then the appropriate logical device is selected for access. Each physical device occupies a number of consecutive byte locations. TABLE 24 sets out the address decoding for a selected number of PnP devices, including Codec 100. For 9-bit decodes A11 . . . A10 are assumed to be zero.

Detailed Description Text (662):
In alternate embodiments, a New Crystal Key may be defined that allows the device to be configured uniquely when two devices coexist in the same system. Microcontroller 103 should support configuring all codec 100 physical/logical devices and downloading of resource data and RAM patch data. In addition a new pin may be defined for providing a "Hardware Strap" function for providing a power-up (RESDRV) defined I/O address for receiving either the Plug-n-Play or Crystal Backdoor Keys. This address replaces the standard 0x279 address. This will enable motherboard devices to be configured through a specific hardware address that is different from the standard PnP address of 0x279. The HWSTRAP pin when pulled low (internal pull-up to VDD) will forge the "Key" address port to one of three fixed addresses. The fixed address is selected by pullups/pulldowns on the HWSTRAP and SCL pins. The use of pin2 (FSYNC) which may be either an input or an output is ok since this pin operates as input when connected to external wavetable, and external wavetahle tri-states, this pin when it is held reset via BRESET.

Detailed Description Paragraph Table (6):
TABLE 5 Register Name Address Register Function Read/Write Mixer Data Latch 0x00 Latches mixer data to ISA bus. Write ISA Data Read 0x00 Read ISA Bus Data Read Sound Blaster Data 0x01 Holds DSP Output Data to be Read/Write Latch read by ISA bus. A read of this address will cause the SB Command busyl bit to be cleared. MPU-401 Receive Data 0x02 Holds data to be read by ISA Read/Write Latch bus. A read of this address will cause the Transmit Buffer Full Flag to be cleared. STATUS 0x03 Current Status of Sound Blaster Read/Only and MPU-401 Handshake bits. Reserved 0x04 Reserved 0x05 Reserved 0x06 Reserved 0x07 SB Busy2 0x08 Reset Sound Blaster Busy2 Write Reserved 0x09 Block Power Down 0x0A Individual Power Down Bits Read/Write Codec 100 Control 0x0B CS4232 Control Base +1 Bits Read/Write Sound Blaster ADPCM 0x0C SB ADPCM Data Read Latch SB Busyl 0x0D Set Sound Blaster Busy Bit Write SB-DRQ Latch 0x0E Reset current pending Sound Read Blaster DMA Request that was set by a write to 8051 address 0x0E. SB-DRQ Latch 0x0E Generate Sound Blaster DMA Write Request and store data in latch. SB-INT 0x0F Generate Sound Blaster Write Interrupt Plug & Play Address 0x10 Stores data written to address Read Only Register 0x279 from ISA bus. Plug & Play Write_Data 0x11 Stores data written to address Read Only Port 0xA79 from ISA bus. Plug & Play Read_Data 0x12 Written by microcontroller 103 Write Only Register in response to a read from the Read_Data_Port address. Plug & Play State 0x13 Defines current Plug & Play Write Only state. Plug & Play 0x14 Control/Status information Read/Write Control/Status I/O Base Address - 0x15 Lower 8 bits of address Write Only Sound System I/O Base Address - 0x16 Upper 4 bits of address Write Only Sound System I/O Base Address - 0x17 Lower 8 bits of address

Write Only Control I/O Base Address - 0x18 Upper 4 bits of address Write Only Control I/O Base Address-Sound 0x19 Lower 8 bits of address Write Only Blaster I/O Base Address-Sound 0x1A Upper 2 bits of address Write Only Blaster I/O Base Address - 0x1B Lower 8 bits of address Write Only Synth I/O Base Address - 0x1C Upper 2 bits of address Write Only Synth I/O Base Address - 0x1D Lower 8 bits of address Write Only MPU-401 I/O Base Address 0x1E Upper 2 bits of address Write Only MPU-401 I/O Base Address - Game 0x1F Lower 8 bits of address Write Only Port I/O Base Address - Game 0x20 Upper 2 bits of address Write Only Port I/O Base Address 0x21 Lower 8 bits of address Write Only 0-CDROM I/O Base Address 0x22 Upper 2 bits of address Write Only 0-CDROM Interrupt Select - 0x23 Bits [3:0] Write Only Synth Interrupt Select - 0x24 Bits [3:0] Write Only Sound Blaster Interrupt Select - 0x25 Bits [3:0] Write Only Sound System Interrupt Select - 0x26 Bits [3:0] Write Only MPU-401 Interrupt Select - CDROM 0x27 Bits [3:0] Write Only Interrupt Select - 0x28 Bits [3:0] Write Only Control DMA Channel Select - 0x29 Bits [2:0] Write Only Sound Blaster DMA Channel Select - 0x2A Bits [2:0] Playback/Capture Write Only Sound System DMA Channel Select - 0x2B Bits [2:0] Capture Write Only Sound System DMA Channel Select - 0x2C Bits [2:0] Write Only CDROM I/O Base Address 1 - 0x2D Lower 8 bits of address Write Only CDROM I/O Base Address 1 - 0x2E Upper 2 bits of address Write Only CDROM Logical Device Activate 0x2F Activate logical device when Write Only bit=1 I/O Base Address - 0x30 Lower 8 bits of address Write Only Modem I/O Base Address - 0x31 Upper 2 bits of address Write Only Modem Address Mask Register - 0x32 Mask used for programmable Write Only CDROM address range Address Mask Register - 0x33 Mask used for programmable Write Only Modem address range Misc. Hardware 0x34 Miscellaneous Hardware Control Write Only Configuration Control Bits Interrupt Select - 0x35 Bits [2:0] Write Only Modem Physical Device 0x36 For auto-power management Read Only Activity Digital Assist 0x37 Auto-Retrigger Enable/Joystick Read/Write Control/Status Status Joystick #1 X 0x38 Joystick Trigger/X Coordinate Read/Write Coordinate Counter Low Byte Joystick #1 X 0x39 X Coordinate Counter High Byte Read Only Coordinate Joystick #1 Y 0x3A X Coordinate Counter Low Byte Read Only Coordinate Joystick #1 Y 0x3B Y Coordinate Counter High Byte Read Only Coordinate Joystick #2 X 0x3C X Coordinate Counter High Byte Read Only Coordinate Joystick #2 X 0x3D X Coordinate Counter High Byte Read Only Coordinate Joystick #2 Y 0x3E X Coordinate Counter Low Byte Read Only Coordinate Joystick #2 Y 0x3F Y Coordinate Counter High Byte Read Only Coordinate Serial Port Control 0x40 Control for bach serial Read/Write interface Bond Out Override 0x41 Bond Out Override bits Read/Write Port 3 Shadow 0x42 Shadow of Port 3 bits for test Read/Write purposes Program RAM 0x4000 1.5 Kbytes Program RAM Read/Write 0x45FF

h e b b cg b cc e

L3: Entry 4 of 20 File: USPT Sep 30, 2003

US-PAT-NO: 6628999
DOCUMENT-IDENTIFIER: US 6628999 B1

TITLE: Single-chip audio system volume control circuitry and methods

DATE-ISSUED: September 30, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Klaas; Jeff	Austin	TX		
Matthews; Phillip	Austin	TX		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Cirrus Logic, Inc.	Austin	TX			02

APPL-NO: 09/ 031112 [PALM]
DATE FILED: February 26, 1998

PARENT-CASE:
CROSS REFERENCE TO RELATED APPLICATIONS This application for patent is related to the following applications for patent: This application is a division of U.S. Ser. No. 08/949,563 entitled "SINGLE-CHIP AUDIO CIRCUITS, METHODS AND SYSTEMS USING THE SAME", filed Oct. 14, 1997; AUDIO SPATIAL ENHANCEMENT CIRCUITRY AND METHODS USING THE SAME, U.S. patent application Ser. No. 08/031,156, filed concurrently herewith; SINGLE-CHIP AUDIO SYSTEM POWER REDUCTION CIRCUITRY AND METHODS, U.S. patent application Ser. No. 08/031,116, filed concurrently herewith; SIGNAL AMPLITUDE CONTROL CIRCUITRY AND METHODS, U.S. patent application Ser. No. 08/031,439, filed concurrently herewith; SINGLE-CHIP AUDIO SYSTEM MIXING CIRCUITRY AND METHODS, U.S. patent application Ser. No. 08/031,447, filed concurrently herewith; and OSCILLATOR START-UP CIRCUITRY AND SYSTEMS AND METHODS USING THE SAME, U.S. patent application Ser. No. 08/031,444, filed concurrently herewith. These application for patent are hereby incorporated by reference in the present disclosure as fully set forth herein.

INT-CL: [07] G06 F 17/00, H03 G 3/00

US-CL-ISSUED: 700/94; 381/104, 381/107
US-CL-CURRENT: 700/94; 381/104, 381/107

FIELD-OF-SEARCH: 700/94, 381/102, 381/104, 381/105, 381/106, 381/107, 381/109, 704/500, 704/501

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

h e b b g e e e f c e e ge

Search Selected | Search ALL | Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4696035	September 1987	Guido et al.	
☐	4700389	October 1987	Kazuaki	
☐	5140283	August 1992	Reed	
☐	5365195	November 1994	Kageyama	330/284
☐	5369311	November 1994	Wang et al.	
☐	5808575	September 1998	Himeno et al.	341/139
☐	6108428	August 2000	Suzuki et al.	381/104
☐	6259957	July 2001	Alexander et al.	381/119

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
60-206209	October 1985	JP	381/107
08250945	July 1997	JP	
WO 94 16538	July 1994	WO	
WO 96 15484	May 1996	WO	

ART-UNIT: 2644

PRIMARY-EXAMINER: Mei; Xu

ATTY-AGENT-FIRM: Murphy, Esq.; James J. Winstead Sechrest & Minick, P.C.

ABSTRACT:

An audio system 100 includes circuitry 116 for generating an audio data stream and circuitry 103 for generating digital words defining a volume control level. System 100 also includes volume control circuitry 5000 for controlling the amplitude of the audio data stream in response to the digital words. Volume control circuitry 5000 includes a master register 5001 for holding digital words received from circuitry for generating 116 and a slave register 5002 for holding a digital word selectively transferred from master register 5001 in response to an enable signal. Output control circuitry 5005, 5006 and 5007 receive the audio data stream and apply a preselected gain defined by the digital word held in slave register 5002. Volume control circuitry 5000 also includes circuitry 5003, 5004 for generating the enable signal when a digital word held in the master register 5001 changes and the audio data stream output from ouput control circuitry 5005, 5006, 5007 reaches a zero crossing.

7 Claims, 266 Drawing figures

L3: Entry 9 of 20 File: USPT Jun 11, 2002

DOCUMENT-IDENTIFIER: US 6405093 B1
TITLE: Signal amplitude control circuitry and methods

Detailed Description Text (110):
Codec 100 operates in conjunction with a number of other associated Plug & Play devices 909 also coupled to the ISA bus of independent blocks 909 that are mapped to the ISA Bus. Each block 909 has associated with it a set of resource requirements and associated configuration registers, organized into groups called physical devices. TABLE 6 below lists the maximum resource requirements for each physical device. The Intel/Microsoft Plug & Play specification organizes devices into logical groupings (logical devices) comprised of one or more physical devices.

Detailed Description Text (189):
As indicated above, Plug-n-Play organizes physical devices into groups of logical devices. A logical device may be comprised of up to four non-contiguous Memory Address ranges, eight non-contiguous I/O Address ranges, two Interrupts, and two DMA channels. Codec 100 only supports I/O, interrupts, and DMA.

Detailed Description Text (190):
Codec 100 has a fixed physical-to-logical device mapping summarized in TABLE 21. The Plug-n-Play resource data must match the Logical-to-Physical device mapping defined in TABLE 20. Controller 103 firmware translates Plug-n-Play logical device configuration cycles into writes of the appropriate hardware configuration registers.

Detailed Description Text (279):
Once a Plug & Play sequence has transpired each logical device, including Codec 100, will have an I/O base address assigned to it. This assigned base address is stored in each I/O base address register. ISA bus address bits A12 . . . A0 are compared with the values stored in the I/O base address registers, and if a match is found, then the appropriate logical device is selected for access. Each physical device occupies a number of consecutive byte locations. TABLE 24 sets out the address decoding for a selected number of PnP devices, including Codec 100. For 9-bit decodes A11 . . . A10 are assumed to be zero.

Detailed Description Text (1610):
Microcontroller 103 should support configuring all codec 100 physical/logical devices and downloading of resource data and RAM patch data. In addition a new pin may be defined for providing a "Hardware Strap" function for providing a power-up (RESDRV) defined I/O address for receiving either the Plug-n-Play or Crystal Backdoor Keys. This address replaces the standard 0x279 address. This will enable motherboard devices to be configured through a specific hardware address that is different from the standard PnP address of 0x279. The HWSTRAP pin when pulled low (internal pull-up to VDD) will force the "Key" address port to one of three fixed addresses. The fixed address is selected by pullups/pulldowns on the HWSTRAP and SCL pins. The use of pin2 (FSYNC) which may be either an input or an output is ok since this pin operates as input when connected to external wavetable, and external wavetable tri-states, this pin when it is held reset via BRESET.

Detailed Description Paragraph Table (6):
TABLE 5 Register Name Address Register Function Read/Write Mixer Data Latch 0x00 Latches mixer data to ISA bus Write ISA Data Read 0x00 Read ISA Bus Data Read Sound Blaster Data 0x01 Holds DSP Output Data to be Read/Write Latch read by ISA bus. A read of this address will cause the SB Command busyl bit to be cleared. MPU-401 Receive Data 0x02 Holds data to te read by ISA Read/Write Latch bus. A read of this address will cause the Transmit Buffer Full Flag to be cleared. STATUS 0x03 Current Status of Sound Blaster Read/Only and MPU-401 Handshake bits. Reserved 0x04 Reserved 0x05 Reserved 0x06 Reserved 0x07 SB Busy2 0x08 Reset Sound Blaster Busy2 Write Reserved 0x09 Block Power Down 0x0A Individual Power Down Bits Read/Write Codec 100 Control 0x0B CS4232 Control Base +1 Bits Read/Write Sound Blaster ADPCM 0x0C SB ADPCM Data Read Latch SB Busyl 0x0D Set Sound Blaster Busy Bit Write SB-DRQ Latch 0x0E Reset current pending Sound Read Blaster DMA Request that was set by a write to 8051 address 0x0E. SB-DRQ Latch 0x0E Generate Sound Blaster DMA Write Request and store data in latch. SB-INT 0x0F Generate Sound Blaster Write Interrupt Plug & Play Address 0x10 Stores data written to address Read Only Register 0x279 from ISA bus. Plug & Play Write_Data 0x11 Stores data written to address Read Only Port 0xA79 from ISA bus. Plug & Play Read_Data 0x12 Written by microcontroller 103 Write Only Register in response to a read from the Read_Data_Port address. Plug & Play State 0x13 Defines current Plug & Play Write Only state. Plug & Play 0x14 Control/Status information Read/Write Control/Status I/O Base Address - 0x15 Lower 8 bits of address Write Only Sound System I/O Base Address - 0x16 Upper 4 bits of address Write Only Sound System I/O Base Address - 0x17 Lower 8 bits of address Write Only Control I/O Base Address - 0x18 Upper 4 bits of address Write Only Control I/O Base Address-Sound 0x19 Lower 8 bits of address Write Only Blaster I/O Base Address-Sound 0x1A Upper 2 bits of address Write Only Blaster I/O Base Address - 0x1B Lower 8 bits of address Write Only Synth I/O Base Address - 0x1C Upper 2 bits of address Write Only Synth I/O Base Address - 0x1D Lower 8 bits of address Write Only MPU-401 I/O Base Address 0x1E Upper 2 bits of address Write Only MPU-401 I/O Base Address - Game 0x1F Lower 8 bits of address Write Only Port I/O Base Address - Game 0x20 Upper 2 bits of address Write Only Port I/O Base Address 0x21 Lower 8 bits of address Write Only O-CDROM I/O Base Address 0x22 Upper 2 bits of address Write Only O-CDROM Interrupt Select - 0x23 Bits [3:0] Write Only Synth Interrupt Select - 0x24 Bits [3:0] Write Only Sound Blaster Interrupt Select - 0x25 Bits [3:0] Write Only Sound System Interrupt Select - 0x26 Bits [3:0] Write Only MPU-401 Interrupt Select - CDROM 0x27 Bits [3:0] Write Only Interrupt Select - 0x28 Bits [3:0] Write Only Control DMA Channel Select - 0x29 Bits [2:0] Write Only Sound Blaster DMA Channel Select - 0x2A Bits [2:0] Playback/Capture Write Only Sound System DMA Channel Select - 0x2B Bits [2:0] Capture Write Only Sound System DMA Channel Select - 0x2C Bits [2:0] Write Only CDROM I/O Base Address 1 - 0x2D Lower 8 bits of address Write Only CDROM L/O Base Address 1 - 0x2E Upper 2 bits of address Write Only CDROM Logical Device Activate 0x2F Activate logical device when Write Only bit = 1 I/O Base Address - 0x30 Lower 8 bits of address Write Only Modem I/O Base Address - 0x31 Upper 2 bits of address Write Only Modem Address Mask Register - 0x32 Mask used fot programmable Write Only CDROM address range Address Mask Register - 0x33 Mask used for programmable Write Only Modem address range Misc. Hardware 0x34 Miscellaneous Hardware Control Write Only Configuration Control Bits Interrupt Select - 0x35 Bit [2:0] Write Only Modem Physical Device 0x36 For auto-power management Read Only Activity Digital Assist 0x37 Auto-Retrigger Enable/Joystick Read/Write Control/Status Status Joystick #1 X 0x38 Joystick Trigger/X Coordinate Read/Wrire Coordinate Counter Low Byte Joystick #1 X 0x39 X Coordinate Counter High Byte Read Only Coordinate Joystick #1 Y 0x3A X Coordinate Counter Low Byte Read Only Coordinate Joystick #1 Y 0x3B Y Coordinate Counter High Byte Read Only Coordinate Joystick #2 X 0x3C X Coordinate Counter High Byte Read Only Coordinate Joystick #2 X 0x3D X Coordinate Counter High Byte Read Only Coordinate Joystick #2 Y 0x3E X Coordinate Counter Low Byte Read Only Coordinate Joystick #2 Y 0x3F Y Coordinate Counter High Byte Read Only Coordinate Serial Port Control 0x40 Control for bach serial Read/Write interface Bond Out Override 0x41 Bond Out Override bits Read/Write Port 3 Shadow 0x42 Shadow of Port 3 bits for test Read/Write purposes Program RAM 0x4000 1.5 Kbytes Program RAM Read/Write 0x45FF

h e b b cg b cc e

First Hit Fwd Refs

Generate Collection Print

L3: Entry 9 of 20 File: USPT Jun 11, 2002

US-PAT-NO: 6405093
DOCUMENT-IDENTIFIER: US 6405093 B1

TITLE: Signal amplitude control circuitry and methods

DATE-ISSUED: June 11, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Malcolm, Jr.; Ronald D.	Austin	TX		
Klaas; Jeff	Austin	TX		
Gentry; Mark	Austin	TX		
Matthews; Phillip	Austin	TX		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Cirrus Logic, Inc.					02

APPL-NO: 09/ 031439 [PALM]
DATE FILED: February 26, 1998

PARENT-CASE:
This is a division of application Ser. No. 08/949,563 filed Oct. 14, 1997 entitled SINGLE-CHIP AUDIO CIRCUITS, METHODS, AND SYSTEMS USING THE SAME. CROSS REFERENCE TO RELATED APPLICATIONS This application for patent is related to the following applications for patent: Pending U.S. patent application Ser. No. 08/949,563 entitled "SINGLE-CHIP AUDIO CIRCUITS, METHODS AND SYSTEMS USING THE SAME", filed Oct. 14, 1997; AUDIO SPATIAL ENHANCEMENT CIRCUITRY AND METHODS USING THE SAME, U.S. patent application Ser. No. 09/031,156, filed concurrently herewith; SINGLE-CHIP AUDIO SYSTEM POWER REDUCTION CIRCUITRY AND METHODS, U.S. patent application Ser. No. 09/031,116, filed concurrently herewith; SINGLE-CHIP AUDIO SYSTEM VOLUME CONTROL CIRCUITRY AND METHODS, U.S. patent application Ser. No. 09/031,122, filed concurrently herewith; SINGLE-CHIP AUDIO SYSTEM MIXING CIRCUITRY AND METHODS, U.S. patent application Ser. No. 09/031,447, filed concurrently herewith; and OSCILLATOR START-UP CIRCUITRY AND SYSTEMS AND METHODS USING THE SAME, U.S. patent application Ser. No. 09/031,444, filed concurrently herewith. These applications for patent are hereby incorporated by reference in the present disclosure as fully set forth herein.

INT-CL: [07] G06 F 17/00, H03 G 3/00

US-CL-ISSUED: 700/94; 381/64, 381/107
US-CL-CURRENT: 700/94; 381/107, 381/64

FIELD-OF-SEARCH: 700/94, 381/102, 381/104, 381/105, 381/106, 381/107, 381/109, 381/119, 381/56

h e b b g e e e f c e e ge

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected | Search ALL | Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4696035	September 1987	Guido et al.	
☐	4700389	October 1987	Kazuaki	
☐	5140283	August 1992	Reed	
☐	5369311	November 1994	Wang et al.	
☐	5615256	March 1997	Yamashita	381/104
☐	5659466	August 1997	Norris et al.	700/94
☐	5835375	November 1998	Kitamura et al.	700/94
☐	6141597	October 2000	Botzko et al.	700/94
☐	6259957	October 2001	Alexander et al.	700/94

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
08250945	July 1997	JP	
WO 94 16538	July 1994	WO	
WO 96 15484	May 1996	WO	

ART-UNIT: 2644

PRIMARY-EXAMINER: Mei; Xu

ATTY-AGENT-FIRM: Murphy; James J. Winstead Sechrest & Minick

ABSTRACT:

Amplitude control circuitry 5000 includes a first register 5001 for storing received amplitude control data and a second register 5002 for storing amplitude control data transfered from first register 5001. Output circuitry 5005, 5006, 5007 controls the amplitude of a received signal in response to amplitude data transferred from second register. A sensor 5003 determines when data stored in first register 5001 and second register 5002 does not match and enables a comparator 5004 when data in first register 5001 and, second register 5002 does not match. Comparator 5004 compares a signal output from output circuitry 5005, 5006, 5007 with a reference signal and generates a signal for enabling the transfer of data from first register 5001 to second register 5002 when the signal output from output circuitry 5005, 5006, 5007 falls within a window defined by the reference voltage.

6 Claims, 263 Drawing figures

h e b b g e e e f c e e ge

L7: Entry 9 of 21 File: USPT Jul 29, 2003

DOCUMENT-IDENTIFIER: US 6601073 B1
TITLE: Deductive database architecture for geographic data

Detailed Description Text (79):
In one embodiment of the deductive database engine, results are obtained and processed one at a time. A query of the data in the geographic database is performed by routines 262 in the index access management system 260 that are called by execution of logic rules 300 by the deductive database engine 200. A query may be performed on all or part of the data in the geographic database, as dictated by the logic rules that define the query. Within a set of data records, the first record in the set that satisfies the query is processed using the appropriate routines 282 in the physical-to-logical conversion system 280. Another logic rule executed by the deductive database engine performs a "GetNext" operation so that after the first record is found that satisfies a query, the query continues from that point forward in the set to find the next record in the set that satisfies the query, and so on, until no more records are found in the set that satisfy the query.

Detailed Description Text (123):
In an alternative embodiment, the logic rules that take into account the hardware resources of the navigation system may be included among the logic rules stored with the geographic database. If the logic rules that take into account the hardware resources of the navigation system are included among the logic rules stored with the geographic database, they call routines in the data access layer that query the hardware of the navigation system in order to determine what resources are available. Based on the responses to these queries, the available hardware resources of the navigation system are taken into account when performing data access functions. Using logic rules and a deductive database engine, a data access layer can be automatically configured for the navigation system in which it is installed (e.g., plug-and-play configuration). In addition, when new hardware resources are added, e.g., more memory, the routines called by the logic rules automatically detect the new hardware and modify the data access functions accordingly.

First Hit Fwd Refs

Generate Collection Print

L7: Entry 9 of 21 File: USPT Jul 29, 2003

US-PAT-NO: 6601073
DOCUMENT-IDENTIFIER: US 6601073 B1

TITLE: Deductive database architecture for geographic data

DATE-ISSUED: July 29, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Robare; Philip	Chicago	IL		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Navigation Technologies Corp.	Chicago	IL			02

APPL-NO: 09/ 532617 [PALM]
DATE FILED: March 22, 2000

INT-CL: [07] G06 F 17/30, G01 C 21/00, G08 G 1/123

US-CL-ISSUED: 707/104; 707/102, 701/202, 701/208, 701/209, 701/210, 340/995
US-CL-CURRENT: 340/995.1, 701/202, 701/208, 701/209, 701/210, 707/102, 707/104.1

FIELD-OF-SEARCH: 707/104, 707/102, 707/200, 707/100, 342/357.13, 701/202, 701/208, 701/209, 701/210, 340/995

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	6047280	April 2000	Ashby et al.	
☐	6208934	March 2001	Bechtolsheim et al.	701/209
☐	6212472	April 2001	Nonaka et al.	701/208
☐	6240361	May 2001	Ise et al.	701/208
☐	6249742	June 2001	Friederich et al.	701/202
☐	6278942	August 2001	McDonough	701/210
☐	6282492	August 2001	Gorai et al.	701/209
☐	6292745	September 2001	Robare	701/208

h e b b g e e e f c e e ge

☐	6308177	October 2001	Israni et al.	707/100
☐	6313761	November 2001	Shinada	340/995
☐	6314367	November 2001	Ohler et al.	701/208
☐	6317753	November 2001	McGrath et al.	707/200
☐	6362779	March 2002	Meek et al.	342/357.13
☐	6370539	April 2002	Ashby et al.	707/102
☐	6381537	April 2002	Chenault et al.	701/209

ART-UNIT: 2171

PRIMARY-EXAMINER: Coby; Frantz

ATTY-AGENT-FIRM: Kozak; Frank J. Shutter; Jon D. Kaplan; Lawrence M.

ABSTRACT:

A database architecture for using geographic data to provide navigation-related functions is disclosed. The navigation-related functions are provided by navigation program applications. A geographic database is stored on a medium and includes data representing geographic features and has a plurality of indexes into the data. A data access layer accepts requests from the navigation program applications for geographically-referenced data, accesses the geographic database and provides responses to the requests from the navigation program applications for geographically-referenced data. Logic rules are associated with the geographic database. The data access layer includes a deductive database engine that accesses and combines the logic rules to determine how to use to the indexes to access the data from the medium and to convert the data from a format in which they are stored on the medium into a format that the navigation program applications can use. The database architecture can be used in vehicle navigation systems including navigation systems that use data obtained via a wireless communications link from an off-board data supplier.

19 Claims, 12 Drawing figures

L7: Entry 12 of 21 File: USPT Oct 1, 2002

DOCUMENT-IDENTIFIER: US 6460043 B1
** See image for Certificate of Correction **
TITLE: Method and apparatus for operating on data with a conceptual data manipulation language

Detailed Description Text (160):
SELECT processing: Referring to FIG. 37, the processing of a SELECT statement will now be described. The CQL statement is first parsed into logical units for further processing, for example with respect to each Fact requested (160). Each requested Fact is scoped (162), as explained above, to determine the Predicate to which it should be bound, and the SID tree is identified (164). The join scope for the Predicate is then identified (166), and the join scope is then reduced to final physical joins in the form of the LTID tree (168). Next, a suitable SQL SELECT statement (SQL being the preferred DML) is ascertained from the LTID tree. This process involves a recursive walk of the tree, beginning from the root with pre-walk and post-walk processing. First, in the prewalk, the basic shape of the SELECT statement is emitted along with the "select list"--the items which will be emitted (170). Each Fact that was present in the CQL statement is mapped to the LTID which it will come from (per the data dictionary information) so this is accomplished with a simple walk through the list of Facts sought in the CQL query. The last preamble task is to emit the name of the table associated with the root of the LTID tree as the first part of the SQL "From" clause (172). This is the physical table that was the root of the CQL query and it will be the root of the SQL From clause. Next, the recursive portion of the SELECT processing begins. Each child of the current LTID is visited and a SQL "join" is formed between the current LTID and the child (174)--the join is a simple INNER JOIN where the LTID corresponds to required Fact, a LEFT OUTER JOIN where the LTID is optional, and either a NOT EXISTS or EXISTS sub-clause where exclusion or inclusion (but not a join) is specified. These join types correspond directly to the [required], [optional], and [exclude] constructs in CQL and indeed survive each successive translation from conceptual to physical. Recalling that the edges in the LTID tree correspond to foreign keys or virtual foreign keys it is readily seen that the join condition (i.e. the linkage between the parent LTID and the child LTID) is equality of all the corresponding columns that are part of the foreign key (remembering that foreign keys constrain columns at one end to be equal to corresponding columns in a unique key on the other end). These operations give the basic shape of the query with all the joins in place. Additionally, as each LTID is visited, any constraints and/or filters associated with that table are emitted into either the FROM, WHERE, or HAVING clause as appropriate for the type of join and the aggregate or non-aggregate nature of the filtered items (176). Once the recursion is complete it remains only to append the accumulated FROM, WHERE, and HAVING components to the preamble (178) and then to generate a suitable GROUP BY clause (180) based on the presence of aggregates (DSL 62 preferably emits a GROUP BY clause that groups by all non-aggregate Facts in the order they occurred) and finally to generate an ORDER BY clause (182) that corresponds to all of the Facts that were found in CQL "sort by" clauses--as with the selected Facts these have already been reduced to columns on particular local tables (i.e. bound to an LTID). It readily follows from the above described process that explicit DELETE, UPDATE and INSERT statements can be generated from explicit CQL for those respective statements in an analogous manner. Rowset based processing of the UPDATE family of commands is discussed below.

Detailed Description Text (179):
The OLE DB interfaces 264 each provide a proven, robust database interface which defines objects and interfaces that are geared for database services. OLE DB componentizes data access providers and allows easy plug and play of components from different sources. One possible OLE DB provider is Microsoft's Microsoft OLE DB Provider for ODBC. By implementing the data services using OLE DB, the system can interface between them in a well-defined manner, allowing flexibility to solve the problems at hand. Any components exposed via OLE DB are accessible in C++ as well as Visual Basic/Script (VB/S) using ADO via a simple yet powerful object model. However, it shall be understood that OLE DB is in no way critical to the invention, and that DSL 62 can be implemented in a variety of other ways that do not use OLE DB.

First Hit Fwd Refs

Generate Collection Print

L7: Entry 12 of 21 File: USPT Oct 1, 2002

US-PAT-NO: 6460043
DOCUMENT-IDENTIFIER: US 6460043 B1
** See image for Certificate of Correction **

TITLE: Method and apparatus for operating on data with a conceptual data manipulation language

DATE-ISSUED: October 1, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Tabbara; Bassam	Seattle	WA		
Mariani; Rico	Kirkland	WA		
Brandes; Kristi L.	Bellevue	WA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Microsoft Corporation	Redmond	WA			02

APPL-NO: 09/ 258417 [PALM]
DATE FILED: February 26, 1999

PARENT-CASE:
CONTINUATION DATA This application is a continuation-in-part of U.S. Ser. No. 09/018,287, now U.S. Pat. No. 6,148,287 entitled "Automatic Generation of Database Queries," and filed Feb. 4, 1998.

INT-CL: [07] G06 F 17/30

US-CL-ISSUED: 707/100; 707/101, 707/3, 707/4
US-CL-CURRENT: 707/100; 707/101, 707/3, 707/4

FIELD-OF-SEARCH: 707/2, 707/3, 707/100, 707/101, 707/4, 707/102

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	5201047	April 1993	Maki et al.	707/4
☐	5377103	December 1994	Lamberti et al.	704/9

h e b b g e e e f c e e ge

☐	5418957	May 1995	Narayan	717/1
☐	5495604	February 1996	Harding et al.	707/102
☐	5519855	May 1996	Neeman et al.	707/3
☐	5548755	August 1996	Leung et al.	707/2
☐	5548770	August 1996	Bridges	707/2
☐	5590319	December 1996	Cohen et al.	707/4
☐	5590321	December 1996	Lin et al.	707/10
☐	5600831	February 1997	Levy et al.	707/2
☐	5615337	March 1997	Zimowski et al.	707/3
☐	5632015	May 1997	Zimowski et al.	707/3
☐	5666526	September 1997	Reiter et al.	707/2
☐	5734887	March 1998	Kingberg et al.	707/100
☐	5806059	September 1998	Tsuchida et al.	707/2
☐	5822747	October 1998	Graefe et al.	707/2
☐	5842212	November 1998	Ballurio et al.	707/100
☐	5857180	January 1999	Hallmark et al.	707/2
☐	5890150	March 1999	Ushijima et al.	707/3
☐	5893095	April 1999	Jain et al.	707/6
☐	5918225	June 1999	White et al.	707/3
☐	5999664	December 1999	Mahoney et al.	382/305
☐	6167405	December 2000	Rosensteel, Jr. et al.	707/102

OTHER PUBLICATIONS

Managaki et al., A Database Design System with Conceptual Model Description Language, IEEE online, pp. 141-146, Nov. 1979.*
Owei et al., Natural Language Query Filtration in the Conceptual Query Language, IEEE online, pp. 539-549, Jan. 1997.*
Jarzabek et al., Model-Based Design of Tools for Business Understanding and Re-Engineering, IEEE online, pp. 328-337, Jul. 1995.

ART-UNIT: 2177

PRIMARY-EXAMINER: Robinson; Greta L.

ATTY-AGENT-FIRM: Lee & Hayes, PLLC

ABSTRACT:

A data services layer is disclosed which maintains a dictionary of conceptual information and physical information about the data. Machine-readable requests to access the data are in a form related to a conceptual organization of the data, and is not specific to a physical organization of the data. A machine-readable query to obtain a subset of the data is produced by referencing the dictionary of conceptual and physical information about the data. The conceptual information is obtained from an object-relational-model of the data, and the physical information indicates

h e b b g e e e f c e e ge

how the data is organized on the data storage medium. Requests are written in a conceptual query language (CQL) which substantially uses terms belonging to or derived from a natural language. CQL includes terms in the classes of names and concepts, and wherein name terms are used to describe objects in the object-relational-model of the data, and concept terms are used to specify the data subset desired. Concept terms specify Facts desired from the data, and filters and sort specifications to be applied to the Facts. In an example embodiment, the data is organized in rows, and CQL includes a select command that retrieves data in rows. A set of data representing a profile of performance characteristics related to how to retrieve data is provided, and queries are formed based at least in part on the performance characteristics.

48 Claims, 67 Drawing figures

L7: Entry 13 of 21 File: USPT Jul 23, 2002

DOCUMENT-IDENTIFIER: US 6425005 B1
TITLE: Method and apparatus for managing local resources at service nodes in an intelligent network

Detailed Description Text (10):
As illustrated and like the "plug and play" feature of personal computers, the ICP software architecture is an open processing model that allows the interchangeability of: (1) management software; (2) ICP applications; (3) computing hardware and software; (4) resource complex components; and even (5) service architecture and processing. Such a generic architecture reduces maintenance costs due to standardization and provides the benefits derived from economies of scale.

Detailed Description Text (59):
After querying its service profile (configuration) file 580 to determine which services are to be immediately instantiated, the NOS LRM component 575 sends a service activation request from NOS NT 570 to the active Service Manager object 554 in SLEE via the NOS client instance 558 also executing in the SLEE 450. The SM object 554 is an API object for enabling control of SLEE services. For example, it provides the capability to instantiate new services when a request for an inactive service is received. That is, it is capable of assigning a process thread to the object when it is instantiated and the service then registers itself with NOS via LRM 575 as depicted generally in FIG. 9. As a service is called by another service, using its logical name, the LRM uses the rules in the configuration file to determine which instance to invoke by utilizing the ORB name service to map the logical name to physical addresses of active instances.

First Hit Fwd Refs

Generate Collection Print

L7: Entry 13 of 21 File: USPT Jul 23, 2002

US-PAT-NO: 6425005
DOCUMENT-IDENTIFIER: US 6425005 B1

TITLE: Method and apparatus for managing local resources at service nodes in an intelligent network

DATE-ISSUED: July 23, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Dugan; Andrew	Superior	CO		
Holmes; Allen	Colorado Springs	CO		
Porter; Kelvin R.	Dallas	TX		
Robb; Terence A.	Colorado Springs	CO		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
MCI WorldCom, Inc.	Washington	DC			02

APPL-NO: 09/ 420654 [PALM]
DATE FILED: October 19, 1999

PARENT-CASE:
CROSS-REFERENCE TO RELATED APPLICATION This Application is a Continuation-In-Part of commonly assigned, co-pending U.S. patent application Ser. No. 09/128,937 filed Aug. 5, 1998, still pending, entitled "Intelligent Call Platform for an Intelligent Network Distributed Architecture" which claims the benefit of U.S. Provisional Application Ser. No. 60/061,173, filed Oct. 6, 1997, now abandoned, both of which are incorporated herein in their entirety by reference thereto. This application additionally claims the benefit of U.S. Provisional Application Ser. No. 60/104,890 filed Oct. 20, 1998, now abandoned, herein incorporated by reference.

INT-CL: [07] G06 F 13/00

US-CL-ISSUED: 709/223; 379/201
US-CL-CURRENT: 709/223; 379/201.01

FIELD-OF-SEARCH: 709/200, 709/201, 709/217, 709/218, 709/219, 709/223, 709/224, 379/201

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

h e b b g e e e f c e e ge

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4713806	December 1987	Oberlander	370/358
☐	5157390	October 1992	Yoshie et al.	340/2.7
☐	5323452	June 1994	Dickman et al.	379/201
☐	5335268	August 1994	Kelly, Jr. et al.	379/112.05
☐	5450480	September 1995	Man et al.	379/201
☐	5463682	October 1995	Fisher et al.	379/201
☐	5475817	December 1995	Waldo et al.	709/316
☐	5537466	July 1996	Taylor et al.	379/201
☐	5551035	August 1996	Arnold et al.	709/315
☐	5619557	April 1997	Van Berkum	379/112
☐	5644629	July 1997	Chow	379/142.07
☐	5664102	September 1997	Faynberg	709/246
☐	5742668	April 1998	Pepe et al.	455/415
☐	5754939	May 1998	Herz et al.	430/5
☐	5799153	August 1998	Blau et al.	709/223
☐	5825865	October 1998	Oberlander et al.	379/211.02
☐	5826268	October 1998	Shaefer et al.	707/201
☐	5838970	November 1998	Thomas	709/316
☐	5867498	February 1999	Gillman et al.	370/385
☐	5881134	March 1999	Foster et al.	379/88.01
☐	5898839	April 1999	Berteau	709/227
☐	5907607	May 1999	Waters et al.	379/230
☐	5915008	June 1999	Dulman	379/221.08
☐	5940616	August 1999	Wang	717/4
☐	5958016	September 1999	Chang et al.	709/229
☐	5966434	October 1999	Schafer	379/201.01
☐	5999965	December 1999	Kelly	709/202
☐	6041109	March 2000	Waller et al.	379/219
☐	6041117	March 2000	Androski	379/268
☐	6208856	March 2001	Jonsson	455/424

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
WO 00/23898	April 2000	WO	
WO 00/24182	April 2000	WO	

h e b b g e e e f c e e ge

ART-UNIT: 2153

PRIMARY-EXAMINER: Meky; Moustafa M.

ABSTRACT:

A resource management system for an intelligent communications network having one or more distributed service nodes, each service node for providing services relating to an event received at a network resource associated with a service node. The system comprising a first processing tier comprising one or more local execution environments located at a node, each execution environment including a mechanism for instantiating one or more service objects capable of performing event services at a first local execution environment, and, for generating status information relating to executing service objects; and, a second processing tier associated with a service node and including a system processor for tracking status and availability of service objects and local execution environments. Upon receipt of service requests, the system processor communicates with the first processing tier for receiving the status information and initiating service object instantiation in the one or more local execution environments in the first processing tier at the node based upon the status and availability information of the requested service object.

36 Claims, 21 Drawing figures

L7: Entry 14 of 21 File: USPT May 21, 2002

DOCUMENT-IDENTIFIER: US 6393481 B1
TITLE: Method and apparatus for providing real-time call processing services in an intelligent network

Detailed Description Text (10):
As illustrated and like the "plug and play" feature of personal computers, the ICP software architecture is an open processing model that allows the interchangeability of: (1) management software; (2) ICP applications; (3) computing hardware and software; (4) resource complex components; and even (5) service architecture and processing. Such a generic architecture reduces maintenance costs due to standardization and provides the benefits derived from economies of scale.

Detailed Description Text (85):
As shown in FIGS. 8-10 the NOS functional sub-components include: 1) a Name Translation ("NT") process 570 that resolves logical names for data and service objects to physical addresses that identifies both the computer (as a network address) and the memory address in which the requested object is running; 2) Local Resource Management ("LRM") processes 575, 577 that tracks and maintains the status of resources executing at the SLEE and at a service node; 3) a global Network Resource Status ("NRS") process 590 as shown and described in greater detail in commonly-owned, co-pending U.S. patent application Ser. No. 09/420,655 filed Oct. 19, 1999, entitled METHOD AND APPARATUS FOR MANAGING RESOURCES IN AN INTELLIGENT NETWORK, the contents and disclosure of which is incorporated by reference as if fully set forth herein, that maintains the status of all service node resources throughout the entire NGIN network and, to provide inter-process communications; 4) a set of services for providing object connectivity, such as that provided by a Common Object Request Broker Architecture (CORBA)-compliant ORB, which enables communications among objects across different computer platforms, API message sets, and Internet Protocol (IP) communications in a manner such as to meet or exceed certain real-time call processing performance requirements. For example, the typical response time for processing a typical 1-800-number "collect call" event, should be approximately 50 to 100 msec.

Detailed Description Text (94):
After querying its service profile (configuration) file 580 to determine which services are to be immediately instantiated, the NOS LRM component 575 sends a service activation request from NOS NT 570 to the active Service Manager object 554 in SLEE via the NOS client instance 558 also executing in the SLEE 450. The SM object 554 is an API object for enabling control of SLEE services. For example, it provides the capability to instantiate new services when a request for an inactive service is received. That is, it is capable of assigning a process thread to the object when it is instantiated and the service then registers itself with NOS via LRM 575. As a service is called by another service, using its logical name, the LRM uses the rules in the configuration file to determine which instance to invoke by utilizing the ORB name service to map the logical name to physical addresses of active instances.

Detailed Description Text (106):
In the context of 1-800- call ("18C") , an 18C call processing and service utilization scenario is now described for exemplary purposes, with reference to the

flow chart of FIGS. 13(a)-13(c) and the conceptual functional diagram of FIG. 18. First, as shown at step 920, a call first arrives at the NGS switch fabric 75. When the NGS receives a call, a bearer control component (FIG. 5) provides the call control component with the access line on which the call was received, as well as the ANI, dialed number, and other data needed for call processing. A call control component maintains a state model for the call, as executed in accordance with its programmed logic. Additionally included in the state model are triggers for instantiating an ELP 540 and sending a service request to a FD 510 as shown in FIG. 18. To instantiate an ELP, the NGS call control component 90 addresses a message to NOS, using a logical name for an ELP, as indicated at step 923, in FIG. 13(a). The NOS, in response, sends a message to a Service Manager object (FIG. 8) to instantiate an ELP within a SLEE and returns an object reference for that ELP back to call control, as depicted in step 926. The NGS call control component includes this object reference in a service request message that is sent to an FD in the SLEE, as indicated at step 929. Thus, all qualified event data that are generated for the call by any process are written to the instantiated ELP process. Particularly, the service request message is addressed to a logical name for FD; this logical name is translated by the NOS NT component to a physical address for an FD logic program that is running at the same service node on which the call was received. Included in the service request message is the dialed number, ANI, and other data.

First Hit Fwd Refs

Generate Collection Print

L7: Entry 14 of 21 File: USPT May 21, 2002

US-PAT-NO: 6393481
DOCUMENT-IDENTIFIER: US 6393481 B1

TITLE: Method and apparatus for providing real-time call processing services in an intelligent network

DATE-ISSUED: May 21, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Deo; Ajay P.	Lewisville	TX		
Syed; Sami	Tervuren			BE
Wang; Henry	Irvine	CA		
Wong; Wendy T.	Dallas	TX		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
WorldCom, Inc.	Clinton	MS			02

APPL-NO: 09/ 421827 [PALM]
DATE FILED: October 20, 1999

PARENT-CASE:
CROSS-REFERENCE TO RELATED APPLICATION This Application is a Continuation-In-Part of commonly assigned, U.S. patent application Ser. No. 09/128,937 filed Aug. 5, 1998 entitled "Intelligent Call Platform for an Intelligent Network Distributed Architecture" which claims the benefit of U.S. Provisional Application Ser. No. 60/061,173, filed Oct. 6, 1997 both of which are incorporated herein in their entirety by reference thereto. This application additionally claims the benefit of U.S. Provisional Application Ser. No. 60/104,890 filed Oct. 20, 1998 the whole contents and disclosure of which is incorporated by reference as if fully set forth herein.

INT-CL: [07] G06 F 15/173

US-CL-ISSUED: 709/224; 709/218, 709/223, 709/224, 709/226, 709/249, 379/201, 379/207, 379/219, 379/220, 379/230, 379/266, 340/201, 340/219, 340/220, 340/522, 340/585
US-CL-CURRENT: 709/224; 340/522, 340/585, 379/201.01, 379/219, 379/220.01, 379/230, 379/266.01, 709/218, 709/223, 709/226, 709/249

FIELD-OF-SEARCH: 370/201, 370/219, 370/220, 370/585, 370/522, 379/201, 379/219, 379/220, 379/266, 379/207, 379/230, 709/224, 709/218, 709/226, 709/223, 709/249

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

h e b b g e e e f c e e ge

Search Selected | Search ALL | Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4713806	December 1987	Oberlander	
☐	5157390	October 1992	Yoshie et al.	
☐	5323452	June 1994	Dickman et al.	
☐	5335268	August 1994	Kelly, Jr. et al.	
☐	5450480	September 1995	Man et al.	
☐	5463682	October 1995	Fisher et al.	
☐	5475817	December 1995	Waldo et al.	
☐	5537466	July 1996	Taylor et al.	
☐	5551035	August 1996	Arnold et al.	
☐	5619557	April 1997	Van Berkum	
☐	5644629	July 1997	Chow	
☐	5664102	September 1997	Faynberg	
☐	5742668	April 1998	Pepe et al.	
☐	5754939	May 1998	Herz et al.	
☐	5799153	August 1998	Blau et al.	
☐	5825865	October 1998	Oberlander et al.	
☐	5826268	October 1998	Shaefer et al.	
☐	5838970	November 1998	Thomas	
☐	5867498	February 1999	Giltman et al.	
☐	5881134	March 1999	Foster et al.	
☐	5898839	April 1999	Berteau	
☐	5907607	May 1999	Waters et al.	
☐	5915008	June 1999	Dulman	
☐	5940616	August 1999	Wang	
☐	5958016	September 1999	Chang et al.	
☐	5966434	October 1999	Schafer	
☐	5999965	December 1999	Kelly	
☐	6041109	March 2000	Waller et al.	
☐	6041117	March 2000	Androski	

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
00/23898	April 2000	WO	

h e b b g e e e f c e e ge

00/24182 April 2000 WO

ART-UNIT: 2152

PRIMARY-EXAMINER: Harrell; Robert B.

ASSISTANT-EXAMINER: Farahi; Farzaneh

ABSTRACT:

System and methodology for providing real-time call processing services received at a switch in an intelligent network having one or more service nodes having originating switches for receiving a call event. The system includes a platform-independent communication system for enabling communication between object instances executing at service nodes in the intelligent network. An operating system agent object instance executing in an execution environment associated with an originating switch communicates call origination information corresponding to a call event received at the switch to one or more object instances executing in an execution environment provided at a service node in the network; the object instances including a line object instance for maintaining the state of a communications line associated with a call origination, and, a service object implementing methods for performing a service according to a customer request. A first database storage device accessible by the service object provides call routing information according to a customer's subscription. A second database storage device is accessible by the service object to provide a corresponding terminating switch location address at a node in the network for the call based on the retrieved call routing information. The platform-independent communication system communicates call routing commands between the service object and at least the line object instance, for enabling call connection between originating and terminating switches independent of their location in the network.

16 Claims, 25 Drawing figures

L7: Entry 15 of 21 File: USPT Mar 26, 2002

DOCUMENT-IDENTIFIER: US 6363411 B1
TITLE: Intelligent network

Detailed Description Text (10):
As illustrated and like the "plug and play" feature of personal computers, the ICP software architecture is an open processing model that allows the interchangeability of: (1) management software; (2) ICP applications; (3) computing hardware and software; (4) resource complex components; and even (5) service architecture and processing. Such a generic architecture reduces maintenance costs due to standardization and provides the benefits derived from economies of scale.

Detailed Description Text (181):
After querying its service profile (configuration) file 580 to determine which services are to be immediately instantiated, the NOS LRM component 575 sends a service activation request from NOS NT 570 to the active Service Manager object 554 in SLEE via the NOS client instance 558 also executing in the SLEE 450. The SM object 554 is an API object for enabling control of SLEE services. For example, it provides the capability to instantiate new services when a request for an inactive service is received. That is, it is capable of assigning a process thread to the object when it is instantiated and the service then registers itself with NOS via LRM 575. As a service is called by another service, using its logical name, the LRM uses the rules in the configuration file to determine which instance to invoke by utilizing the ORB name service to map the logical name to physical addresses of active instances.

Detailed Description Text (232):
FIG. 18(d) is a sequence diagram describing the steps for instantiating an SLP relating to a received service request (as indicated at step 1028b, FIG. 18(a)). Preferably, a request for multiple SLPs may be made in a single request such that the SLP, CLP and LLPO corresponding to the requested call service may be instantiated concurrently. Utilizing the results of the FD DB query, [step 1025, FIG. 18(a)], the FD SLP sends the SLP logical name to NT as indicated at step 1050, FIG. 18(d) and NT, in turn, queries its instance tables, e.g., local DM cache for the name translation for the physical location (object reference) of the SLP to execute as indicated at step 1051. The DM (local cache) sends back the object reference of the SLP(s) (storage address), as indicated at step 1052. The NT then queries the NNOS LRM to find out if the SLP is instantiated locally and, if not, which instance of the requested service to use, as indicated at step 1053. In response, the LRM returns the actual SLP name with the SLEE addresses at step 1054. The NNOS, in response, may send a request to the Service Manager object running on a Service Control SLEE in order to instantiate a new SLP service, or alternately, request that the service=s thread manager assign a new thread for the requested service having a unique tracking identifier representing the call. In the preferred embodiment, NNOS will select the SLP from a Service Control server that received the original incoming service request notification from the NGS, however, it is understood that NNOS could select the SLP in any service control component through implementation of the NNOS LRM and the NRS list of Service Control instances and their current status. The next step of FIG. 18(d), requires that the instantiated SLP process registers its physical address with the NNOS, and that the NNOS allocates this SLP to the service request. Then, at step 1055, the NNOS passes the

service request hand-off message to the new SLP so that the SLP may begin processing the call in accordance with its programmed logic. Parallel to the SLP instantiation process, the associated CLP (and any other SLP) for this call may be instantiated as well, and it should be understood that an ELP instance for this call has been pre-instantiated for call context data collection. Finally, as indicated at step 1057a, FIG. 18(d), the SLP communicates with the CLP providing it with the addresses of the SLP, LLP and the ELP, and at step 1057b, the SLP communicates with the ELP providing it with the addresses of the SLP, LLP and the CLP. Via the CORBA implementation NNOS, interfaces are thus established between the LLP, CLP, SLP.

Detailed Description Text (235):
FIG. 18(e) illustrates the process for instantiating the terminating LLP at a remote NGIN node prior to routing a call. As shown at step 1070, this requires the CLP to send the terminating node location and the logical name of the terminating LLP to NT so that it may be instantiated (the terminating node location is part of the routing response returned from DM). The NT then sends the LLP logical name to DM at step 1071 which returns the actual LLP name plus the addresses of its stored location (object reference) at step 1072. At step 1073, the NT then queries the NNOS NRS function to determine if the node to which this call is terminating is up and operational, and, at step 1074, the NRS returns to NT the status of the terminating node. Via NNOS, the NT of the local node requests the NNOS NT agent of the remote node to instantiate the terminating LLP at step 1075. As indicated at step 1076, this requires the NT on the terminating node to query its LRM to determine if the LLP is already instantiated for this terminating line, and if not, instantiates the LLP. The LRM at the terminating node returns to NT the SLEE address where the LLP for the terminating line is running at step 1077. Then, at step 1078, the NT of the terminating node sends the call data to the LLP of the terminating line and additionally sends the address of the SLEE executing the LLP for the terminating line to the NT of the originating node as indicated at step 1079. The NT of the originating node sends the address of the SLEE executing the LLP for the terminating line to the CLP at step 1080, and, as indicated at step 1081, a local database lookup is performed to determine the features (if any) on the terminating line. Specifically, the terminating LLP sends logical database name of the line info database to NT for name translation. NT requests the actual line information database name from DM and sends the actual line information DB name and its stored locations to NT. NT queries LRM to find out if the line information DB is available locally and LRM sends back the physical DB address to NT. NT passes the line information DB physical address to the terminating LLP. Then, the terminating LLP sends request to DM to look up customer terminating line information and DM returns the customer line information to LLPT. The system is now ready to perform the routing of the call, as will be described.

Detailed Description Text (352):
In the context of ATM Vnet services ("ATM/Vnet"), a processing and service utilization scenario is now described for exemplary purposes, with reference to the functional flow diagrams of FIGS. 32(a)-32(g). First, as shown in FIG. 31(b), an ATM/Vnet call event first arrives at the NGS switch fabric of the NGS 180. When the NGS 180 receives a call, the bearer control component provides the call control component with the access line on which the call was received, as well as the Vnet #, ANI, line ID, Network Call ID, originating switch trunk, and other data needed for call processing. The NGS Call control maintains a state model for the call, as executed in accordance with its programmed logic. Additionally included in the state model are triggers for instantiating the ELP 540 and sending a service request to a FD 510 as shown in FIG. 31(b). To instantiate an ELP, the NGS call control component addresses a message to NNOS, using a logical name for an ELP as described herein. The NNOS, in response, sends a message to a Service Manager object to instantiate an ELP within a SLEE and returns an object reference for that ELP back to call control. The NGS call control component includes this object reference in a service request message that is sent to an FD in the SLEE. Thus, all

qualified event data that are generated for the call by any process are written to the instantiated ELP process. Particularly, the service request message is addressed to a logical name for FD; this logical name is translated by the NNOS NT component to a physical address for an FD logic program that is running at the same service node on which the call was received. Included in the service request message is the Vnet #, ANI, and other data.

Detailed Description Text (367):
It should be understood that, in the context of an ATM to ATM call, no number translation need be performed. For other types of Vnet calls, however, if a number translation is required, the ATM_Vnet process requests that NNOS return an object reference to the Vnet number translation database provided by DM. Once the SLP receives the location of the database, a database query is performed to lookup the physical address associated with the logical destination Vnet number and DM returns the physical address. Accordingly, a terminating profile is used to determine if the destination address can handle ATM and the specified bandwidth. The Vnet number translation may then be written to the ELP instance for placement in DM=s allocated call context database.

First Hit Fwd Refs

Generate Collection Print

L7: Entry 15 of 21 File: USPT Mar 26, 2002

US-PAT-NO: 6363411
DOCUMENT-IDENTIFIER: US 6363411 B1

TITLE: Intelligent network

DATE-ISSUED: March 26, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Dugan; Andrew	Superior	CO		
Holmes; Allen M.	Colorado Springs	CO		
Robb; Terence A.	Colorado Springs	CO		
Deo; Ajay P.	Lewisville	TX		
Syed; Sami	Tervuren			BE
Wong; Wendy T.	Dallas	TX		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
MCI WorldCom, Inc.	Jackson	MS			02

APPL-NO: 09/ 420666 [PALM]
DATE FILED: October 19, 1999

PARENT-CASE:
CROSS-REFERENCE TO RELATED APPLICATION This Application is a Continuation-In-Part of commonly assigned, co-pending U.S. patent application Ser. No. 09/128,937 filed Aug. 5, 1998, entitled "Intelligent Call Platform for an Intelligent Network Distributed Architecture" which claims the benefit of U.S. Provisional Application Ser. No. 60/061,173, filed Oct. 6, 1997, now abandoned. This application additionally claims the benefit of U.S. Provisional Application Ser. No. 60/104,890 filed Oct. 20, 1998, now abandoned.

INT-CL: [07] G06 F 13/00

US-CL-ISSUED: 709/202; 379/201.01
US-CL-CURRENT: 709/202; 379/201.01

FIELD-OF-SEARCH: 709/200, 709/201, 709/202, 709/203, 709/217, 709/218, 709/219, 709/223, 709/224, 379/201.01, 379/201.02, 379/201.03

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

h e b b g eeef c e e ge

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4713806	December 1987	Oberlander	370/358
☐	5157390	October 1992	Yoshie et al.	340/825.52
☐	5323452	June 1994	Dickman et al.	379/201.04
☐	5335268	August 1994	Kelly, Jr. et al.	379/112.05
☐	5450480	September 1995	Man et al.	379/201.03
☐	5463682	October 1995	Fisher et al.	379/209.04
☐	5475817	December 1995	Waldo et al.	709/316
☐	5537466	July 1996	Taylor et al.	379/221.11
☐	5551035	August 1996	Arnold et al.	709/315
☐	5619557	April 1997	Van Berkum	379/265.02
☐	5644629	July 1997	Chow	379/142.07
☐	5664102	September 1997	Faynberg	709/246
☐	5742668	April 1998	Pepe et al.	455/415
☐	5754939	May 1998	Herz et al.	430/5
☐	5799153	August 1998	Blau et al.	709/223
☐	5825865	October 1998	Oberlander et al.	379/211.02
☐	5826268	October 1998	Shaefer et al.	707/9
☐	5838970	November 1998	Thomas	709/316
☐	5867498	February 1999	Gillman et al.	370/385
☐	5881134	March 1999	Foster et al.	379/88.01
☐	5898839	April 1999	Berteau	709/227
☐	5907607	May 1999	Waters et al.	379/201.03
☐	5915008	June 1999	Dulman	379/221.08
☐	5940616	August 1999	Wang	717/34
☐	5958016	September 1999	Chang et al.	709/229
☐	5966434	October 1999	Schafer	379/207.01
☐	5999965	December 1999	Kelly	709/202
☐	6041117	March 2000	Androski	379/260
☐	6208856	March 2001	Jonsson	455/424

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
00/23898	April 2000	WO	
00/24182	April 2000	WO	

ART-UNIT: 2153

h e b b g e e e f c e e ge

PRIMARY-EXAMINER: Meky; Moustafa M.

ABSTRACT:

In a telecommunications switching network having a resource complex including network switches, an intelligent service platform for providing intelligent call processing and service execution for call events received at the switches and requiring call processing services. A centralized administration system is provided that comprises a system for storing one or more reusable business objects that each encapsulate a distinct call-processing function, and any associated data required by the business object; a system for distributing selected business objects and associated data to selected nodes in the switching network based on pre-determined node configuration criteria; and, a system for activating the business objects in preparation for real-time use. A computing platform is provided within each node for executing those business objects required to perform a service in accordance with an event received at the network switch. Also within a node is a storage and retrieval system for sorting and retrieving selected objects and any associated data distributed by the administration system, and making them locally available to the computing platform when required to perform a service. An underlying location-independent communication system is provided to coordinate interaction of one or more business objects to perform the service in response to needs of the received event.

60 Claims, 83 Drawing figures

L7: Entry 16 of 21 File: USPT Jan 1, 2002

DOCUMENT-IDENTIFIER: US 6335927 B1
TITLE: System and method for providing requested quality of service in a hybrid network

Detailed Description Text (1454):
When the switch 221 of FIG. 1D, is connected to the internet 295, the processing is provided as follows. A line from the internet 295 enters the switch through a modem port 268 and enters the pooled switch matrix where demux and other necessary operations are performed before the information is passed to the switch 221 through the internal network 237 and the message bus 234. The modules 261-268 provide plug and play capability for attaching peripherals from various communication disciplines.

Detailed Description Text (1691):
A system and method for providing enhanced SS7 network management functions are provided by a distributed client/server platform that can receive and process events that are generated by various SS7 network elements. Each network event is parsed and standardized to allow for the processing of events generated by any type of element. Events can also be received by network topology databases, transmission network management systems, network maintenance schedules, and system users. Referring to FIG. 3, the systems architecture of the preferred embodiment of the present invention, referred to as an SS7 Network Management System (SNMS), is illustrated. SNMS consists of four logical servers 302/304/306/308 and a plurality of client workstations 312a/312b/312c connected via a Network Management Wide Area Network (WAN) 310. The four logical SNMS servers 302/304/306/308 may all reside on a single or a plurality of physical units. In the preferred embodiment, each logical server resides on a distinct physical unit, for the purpose of enhancing performance. These physical units may be of any conventional type, such as IBM RS6000 units running with AIX operating system.

First Hit Fwd Refs

Generate Collection Print

L7: Entry 16 of 21 File: USPT Jan 1, 2002

US-PAT-NO: 6335927
DOCUMENT-IDENTIFIER: US 6335927 B1

TITLE: System and method for providing requested quality of service in a hybrid network

DATE-ISSUED: January 1, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Elliott; Isaac K.	Colorado Springs	CO		
Reynolds; Tim E.	Iowa City	IA		
Krishnaswamy; Sridhar	Cedar Rapid	IA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
MCI Communications Corporation	Washington	DC			02

APPL-NO: 08/ 751917 [PALM]
DATE FILED: November 18, 1996

INT-CL: [07] H04 L 12/64

US-CL-ISSUED: 370/352
US-CL-CURRENT: 370/352

FIELD-OF-SEARCH: 370/352, 370/353, 370/354, 370/355, 370/356, 370/400, 370/410, 370/431, 370/468, 379/93.01, 379/88.17

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4054756	October 1977	Comella	
☐	4100377	July 1978	Flanagan	
☐	4653045	March 1987	Stanley	
☐	4771425	September 1988	Baran et al.	
☐	4907724	March 1990	Nomura	
☐	4969184	November 1990	Gordon et al.	

h e b b g e e e f c e e ge

☐	4979206	December 1990	Dadden	
☐	4996707	February 1991	O'Malley	
☐	5029196	July 1991	Morganstein	
☐	5068888	November 1991	Scherk	
☐	5115495	May 1992	Tsuchiya et al.	709/239
☐	5146488	September 1992	Okada	
☐	5159624	October 1992	Makita	
☐	5193110	March 1993	Jones	
☐	5204894	April 1993	Dadden	
☐	5247571	September 1993	Kay	
☐	5268957	December 1993	Albrecht	
☐	5287199	February 1994	Zoccolillo	
☐	5311583	May 1994	Friedes	
☐	5396542	March 1995	Alger	
☐	5402478	March 1995	Hiluchyj et al.	379/221
☐	5406557	April 1995	Baudion	
☐	5425091	June 1995	Josephs	
☐	5428608	June 1995	Freeman	
☐	5440620	August 1995	Slusky	
☐	5448633	September 1995	Jamaleddin	
☐	5450411	September 1995	Hell	
☐	5452289	September 1995	Sharma	
☐	5459775	October 1995	Isono	
☐	5463677	October 1995	Bush	
☐	5473608	December 1995	Gagne	
☐	5477531	December 1995	McKee et al.	
☐	5483586	January 1996	Sussman	
☐	5483587	January 1996	Hogan	
☐	5495521	February 1996	Rangachar	
☐	5511111	April 1996	Serbetcioglu	
☐	5521719	May 1996	Yamada	
☐	5521924	May 1996	Kakuma	
☐	5524137	June 1996	Rhee	
☐	5526353	June 1996	Henley et al.	
☐	5526416	June 1996	Dezonno et al.	
☐	5539884	July 1996	Robrock, II	
☐	5541917	July 1996	Farris	
	5541927	July 1996	Kristol	

h e b b g e e e f c e e ge

5541930	July 1996	Klingman
5551025	August 1996	O'Reilly
5559721	September 1996	Ishii
5561670	October 1996	Hoffert
5563882	October 1996	Brund
5579472	November 1996	Keyworth, II
5590127	December 1996	Bales
5590181	December 1996	Hogan et al.
5604682	February 1997	McLaughlin
5604737	February 1997	Iwami et al.
5608786	March 1997	Gordon
5610910	March 1997	Focsaneanu et al.
5617422	April 1997	Litzenberger et al.
5619555	April 1997	Fenton
5623601	April 1997	Vu
5625404	April 1997	Grady
5625407	April 1997	Biggs
5625677	April 1997	Feiertag
5625682	April 1997	Gray
5627886	May 1997	Bowman
5633916	May 1997	Goldhagen
5636216	June 1997	Fox et al.
5646982	July 1997	Hogan et al.
5651006	July 1997	Fujino
5652787	July 1997	O'Kelly
5654250	August 1997	Park
5654957	August 1997	Koyama
5657250	August 1997	Park
5661790	August 1997	Hsu
5661791	August 1997	Parker
5668857	September 1997	McHale
5673263	September 1997	Basso et al.
5675507	October 1997	Bobo, II
5675741	October 1997	Aggarawai
5680392	October 1997	Semaan
5689550	November 1997	Garson et al.
5689553	November 1997	Ahuja

5692039	November 1997	Brankley
5695507	December 1997	Auth et al.
5699089	December 1997	Murray
5699352	December 1997	Kriete
5701295	December 1997	Bales
5703935	December 1997	Raissyan
5703942	December 1997	Pinard
5710884	January 1998	Dedrick
5712903	January 1998	Bartholomew
5712906	January 1998	Grady
5712907	January 1998	Wegner et al.
5724355	March 1998	Bruno et al.
5724412	March 1998	Srinivasan
5726984	March 1998	Kubler
5727129	March 1998	Barrett
5729544	March 1998	Lev
5729599	March 1998	Plomondon
5732078	March 1998	Arango
5737333	April 1998	Civanlar et al.
5737395	April 1998	Irribarren
5737701	April 1998	Rosenthal
5740230	April 1998	Vaudren
5740231	April 1998	Cohn
5742668	April 1998	Pepe et al.
5742670	April 1998	Bennet
5742674	April 1998	Jain
5742762	April 1998	Scholl et al.
5742905	April 1998	Pepe et al.
5745556	April 1998	Ronen
5745702	April 1998	Morozumi
5749075	May 1998	Toader
5751338	May 1998	Ludwig
5751706	May 1998	Land et al.
5751791	May 1998	Chen et al.
5764736	June 1998	Shachar et al.
5764745	June 1998	Chan
5764756	June 1998	Onweller

☐			
☐	5764916	June 1998	Busey et al.
☐	5768513	June 1998	Kuthyar et al.
☐	5768527	June 1998	Zhuet
☐	5781620	July 1998	Montgomery
☐	5782642	July 1998	Goren
☐	5784443	July 1998	Chapman
☐	5784561	July 1998	Bruno
☐	5787150	July 1998	Reiman
☐	5790174	August 1998	Richard, III
☐	5790548	August 1998	Sistanizadeh
☐	5790645	August 1998	Fawcett et al.
☐	5793498	August 1998	Scholl
☐	5799016	August 1998	Onweller
☐	5799307	August 1998	Buitron
☐	5802283	September 1998	Grady
☐	5802510	September 1998	Jones
☐	5802518	September 1998	Karaey
☐	5805587	September 1998	Norris
☐	5809415	September 1998	Rossman
☐	5812278	September 1998	Toyoda
☐	5812654	September 1998	Anderson
☐	5813006	September 1998	Polnerow et al.
☐	5818836	October 1998	Duval
☐	5828370	October 1998	Moeller
☐	5828837	October 1998	Eikeland
☐	5835579	November 1998	Geisi
☐	5835720	November 1998	Nelson
☐	5838682	November 1998	Dekelbaum
☐	5838683	November 1998	Corley
☐	5839063	November 1998	Lee
☐	5844600	December 1998	Kerr
☐	5844972	December 1998	Jagadish
☐	5848143	December 1998	Andrews et al.
☐	5848415	December 1998	Guck
☐	5850388	December 1998	Anderson et al.
☐	5850433	December 1998	Rondeau
	5850442	December 1998	Muffie

h e b b g e e e f c e e ge

☐				
☐	5854893	December 1998	Ludwig	
☐	5856974	January 1999	Gervais	
☐	5859967	January 1999	Kaudeld	
☐	5862203	January 1999	Wulkan et al.	
☐	5862223	January 1999	Walker	
☐	5862325	January 1999	Reed	
☐	5864609	January 1999	Cross	
☐	5865223	February 1999	Walker	
☐	5867562	February 1999	Scherer	
☐	5867571	February 1999	Borchering	
☐	5870557	February 1999	Bellovin	
☐	5872926	February 1999	Levac et al.	
☐	5873077	February 1999	Kanoh	
☐	5873080	February 1999	Coden	
☐	5881064	March 1999	Lin	
☐	5883891	March 1999	Williams	
☐	5884032	March 1999	Bateman et al.	
☐	5884262	March 1999	Wise	
☐	5892764	April 1999	Riemann	
☐	5892924	April 1999	Lyon et al.	
☐	5905736	May 1999	Ronen	
☐	5905777	May 1999	Foladore et al.	
☐	5905862	May 1999	Hoekstra	
☐	5905871	May 1999	Buskens	
☐	5905872	May 1999	DeSimone	
☐	5907547	May 1999	Foladare	
☐	5907602	May 1999	Peel	
☐	5907607	May 1999	Waters	
☐	5915008	June 1999	Dulman	
☐	5923659	July 1999	Curry	
☐	5931961	August 1999	Ranganathan et al.	
☐	5940479	August 1999	Huy et al.	379/93.01
☐	5946299	August 1999	Blonder	
☐	5959996	September 1999	Byers	
☐	5970059	October 1999	Ahopelto	
☐	5970477	October 1999	Roden	
	5999956	December 1999	Kelly	

☐				
☐	6003030	December 1999	Kenner et al.	707/10
☐	6009469	December 1999	Mattaway et al.	
☐	6011794	January 2000	Mordowitz et al.	
☐	6016307	January 2000	Kaplan et al.	370/238
☐	6020915	February 2000	Bruno et al.	
☐	6031904	February 2000	An	
☐	6064653	May 2000	Farris	
☐	6069890	May 2000	White et al.	
☐	6154744	November 2000	Kenner et al.	707/10
☐	6175870	January 2001	Gawlick et al.	709/227

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
0767568	April 1997	EP	
0781016	June 1997	EP	
0802690	October 1997	EP	
09168051	June 1997	JP	
09168063	June 1997	JP	
09168064	June 1997	JP	
09168065	June 1997	JP	
09172459	June 1997	JP	
09172462	June 1997	JP	
9834391 A 3	August 1988	WO	
9107839	May 1991	WO	
9522221	August 1995	WO	
9620553	July 1996	WO	
9632800	October 1996	WO	
9634341	October 1996	WO	
9638018	November 1996	WO	
9714238	April 1997	WO	
9722211	June 1997	WO	
9723078	June 1997	WO	
9728628	August 1997	WO	
9733412	September 1997	WO	
9812860	March 1998	WO	
9823080	May 1998	WO	
9834391 A 1	August 1998	WO	
9834391 A 2	August 1998	WO	

OTHER PUBLICATIONS

Gralla, Preston: "How The Internet Works" Communicating On The Internet, Chapter

h e b b g e e e f c e e ge

12, pp. 64-67.
Miller, Mark: "Managing the Internet", Troubleshooting TCP/IP, Chapter 7, 1992, pp. 365-375.
Oppen, et al.: "The Clearinghouse: A Decentralized Agent For Locating Named Objects In A Distributed Environment" ACM Transactions on Office Information Systems, vol. 1, No. 3, Jul. 1983, pp. 230-253.
C. Yang: "INETPhone: Telephone Services and Servers on Internet," Network Working Group, Apr. 1995, pp. 1-6.
Sriram, Kotikalapudi et al., "Voice Packerization and Compression in Broadband ATM Networks," Apr. 1991, IEEE Journal on Selected Areas in Communications, vol. 9, No. 3, pp. 294-304.
"'Net Telephony Spec Recommended," Communications Week, Mar. 17, 1997, p. 12.
"Internet by Satellite".
"Telephony on the Internet" (Workshop Information) presented by International Quality & Productivity Center, IMTC, and VOICE Technology & Services News, Sep. 26, 1996.
Abstract for 09168033 A, Patent Abstracts of Japan, 1997.
Abstract for 09168051 A, Patent Abstracts of Japan, 1997.
Abstract for 09168063 A, Patent Abstracts of Japan, 1997.
Abstract for 09168064 A, Patent Abstracts of Japan, 1997.
Abstract for 09168065 A, Patent Abstracts of Japan, 1997.
Abstract for 09172459 A, Patent Abstracts of Japan, 1997.
Abstract for 09172462 A, Patent Abstracts of Japan, 1997.
Aidarous et al., "The Role of the Element Management Layer in Network Management" Feb., 1994: pp. 59-69.
Bethony, Herb "HAHTSite Gives Pros Everything They Need," Mar., 1997, pp. 36-37.
Black, V. "OSI: A Model for Computer Communications Standards," Prentice-Hall, Inc. pp. 162-163, 1991.
Bohn, R. et al., "Mitigating the Coming Internet Crunch: Multiple Service Levels via Precedence," Journal of High Speed Network, vol. 3, No. 4, 1994, pp. 335-349.
Braun et al., "Implementation of an Internet Video Conferencing Application Over ATM," IEEE, 1997.
Chen et al., "ATM and Satellite Distribution of Multimedia Educational Courseware," Jun. 1996; pp. 1133-1137.
Civanlar et al., "FusionNet: Joining the Internet & Phone Networks for Multimedia Applications," 1996, pp. 431-432.
Cobbold et al., "Enhancement for Integrated Wireless Personal Communications Over metropolitan Area Networks," Apr., 1996: pp. 1370-1376.
Comer, Douglas, "Internetworking With TCP/IP vol. I: Principles, Protocols, and Architecture"; Third Ed.; Prentice Hall; 1995 pp. 143-153.
Diehls, "Data's New Voice," BYTE Sep. 1996 pp. 129-135.
Duan et al., "Efficient Utilization of Multiple Channels Between Two Switches in ATM Networks," Feb., 1995: pp. 1906-1911.
Ejiri, Masayoshi, "For Whom the Advancing Service/Network Management," Feb., 1994: pp. 442-433.
Elia et al., "Skyplex: Distribution Up-link for Digital Television via Satellite," Dec., 1996: pp. 305-313.
Ely, Tom, "The Service Control Point as a Cross Network Integrator," Apr., 1996: pp. 1-8.
Eriksson, Hans, "MBONE: The Multicase Backbone," Communications of the ACM, vol. 57, No. 8, Aug. 1994, pp. 54-60.
Feinmann, "VIC Computer Telephony," Computer Telephony, Mar. 1996, pp. 219-221.
Fridisch et al, "Terminals for Accessing the Internet--The Internet Telephone," Alcatel Telecommunications Review--4th Quarter 1996, pp. 304-309.
Ganor, Elon, "Talk Talk," Tele.com, Jun. 1996, pp. 68-72.
Gareiss, Robin "Voice Over the Internet," Sep., 1996 Data Communications, vol. 25, No. 12, Sep. 1996, pp. 93, 94, 96, 98, 100.
Gralla, Preston, "How to Make Phone Call: How the Internet Works" Part 4, Chap. 21: pp. 118-119.
Gralla, Preston, "How The Internet Works" Communicating On The Internet, Chapter

12, pp. 64-67 1996.
Grami et al., "The Role of Satellites in the Information Superhighway," Jun., 1995: pp. 1577-1581.
Gupta et al., "Technical Assessment of (T) INA-TMN-OSI Technology for Service Management Applications," Feb. 1994: pp. 877-887.
Gys, L. et al., "Intelligence in the Network" Alcatel Telecommunications Review, No. 1, 1998, pp. 13-22.
Hurwicz, Michael, "Faster Smarter Nets," Apr., 1997: pp. 83-89.
Inamori et al., "Applying TMN to a Distributed Communications Node System with Common Platform Software," Feb., 1995: pp. 83-87.
Jain, Surinder, "Evolving Existing Narrowband Networks Broadband Networks with IN Capabilities," Apr., 1996.
Kahn, Jeffery, "Videoconferencing Debuts on the Internet," Feb. 28, 1995.
Kelly, Katy, "Mountaintop Office Keeps Skiers in Touch," USA Today vol. 15, No. 112.
Kim, Gary, "Talk is Cheap," America's Network, Jul. 15, 1996: pp. 34-39.
Kishimoto, Royozo, "Agent Communication System for Multimedia Communication Services," Mar., 1996: pp. 10-17.
Kolarov et al., "End-to-End Adaptive Rate Based Congestion Control Scheme for ABR Service in Wide Area ATM Networks," Feb., 1995: pp. 138-143.
Kumar, Vinay, "Internet Multicasting: Internet's Next Big Thing," ICAST Corp. 1997.

Lapolla, Stephanie, PC Week, "Net Call Centers, Voice to Merge", Mar. 31, 1997, p. 10.
Li, C. et al., "Time-Driven Priority Flow Control for Real-Time Heterogeneous Internetworking," Proceedings in Computer Communications, Fifteenth Annual Joint Conference of the IEEE Computer and Communication Generation, San Francisco, Mar. 24-28, 1996, vol. 1 Conf., Mar. 24, 1996, IEEE, pp. 189-197.
Louth, Nick, Reuters, "MCI Communications Corporation Vaults Phone-Data Divide" MCI Communications Corp. news page, Jan. 29, 1997, web page attached.
Low, C. et al., "Webin--An Architecture For Fast Development of In-Based Personal Services" Workshop Record Intelligent Network. Freedom and Flexibility: Realizing the Promise of Intelligent Network Services, Apr. 21, 1996, pp. 1-12.
Macedonia et al., "Mbone Provides Audio and Video Across the Internet," Apr. 1994, pp. 30-36.
Margulies, Ed, "CT's Cyberdate with The 'Net," Aug. 1996, Computer Telephony Periscope, pp. 28-29.
Matta, I., et al., "Proceedings of the Confe rence on Computer Communications" IEEE, issue 3, 1994, pp. 492-499.
Miller, Mark, "Troubleshooting TCP/IP," Managing the Internet, Chapter 7, 1992, pp. 365-375.
Newton, Harry, "The Personal Side of CT" Computer Telephony Jan., 1997, pp. 12-14.
Oppen et al.: "The Clearinghouse: A Decentralized Agent For Locating Named Objects In A Distributed Environment" ACM Transactions on Office Information Systems, vol. 1, No. 3, Jul. 1983, pp. 230-253.
Peeren, Rene "The Intelligent Web," Apr., IEE 1996: vol. 1.
Perret et al., "MAP: Mobile Assistant Programming for Large Scale Communications Networks," Apr., 1996: pp. 1128-1132.
Pezzutti, David, "Operations Issued for Advanced Intelligent Networks," IEEE Communications Magazine, Feb. 1992, pp. 58-63.
Platt, Richard, "Why IsoEthernet Will Change the Voice and Video Worlds," IEE Communications Magazine, Apr., 1996, pp. 55-59.
Retkwa, Rosalyn "Telephone Politics," Internet World, Jun. 1996, pp. 54-60.
Schreyer et al., "Least Cost Call Routing," Apr., 1996: pp. 12-17.
Schulzrinne, Henning, "RFC 1889-RTP: A Transport Protocol for Real-Time Applications," Jan. 1996.
Schulzrinne, Henning, "RFC 1890-RTP: Profile for Audio and Video Conferences with Minimal Control," Jan., 1996.
Sclavos, et al., "Information Model: From Abstraction to Application," Feb., 1994: pp. 183-195.

Serrano, Inma R., "Evolution of Hybrid Fibre Coaxial Network for Multimedia Interactive Services," British Telecommunications Engineering, Oct., 1996, pp. 249-253.
Sharp, C. D.and K. Clegg, "Advanced Intelligent Networks--Now a Reality," Electronics & Communication Engineering Journal, Jun. 1994, pp. 153-162.
Sullivan, K. B., "Videoconferencing Arrives on the Internet," Aug. 1996.
Sunaga et al., "A Reliable Communication Switching Platform for Quick Provisioning," Feb., 1995: pp. 77-82.
The Wall Street Journal, "MCI's New Service for Corporate Use Sets 1 Line for Net, Phone", Jan. 30, 1997, web page attached.
Tsuchida et al., "Intelligent Dynamic Service Provisioning Architecture in the Multimedia Era," Communications--Gateway to Globalization. Apr., 1996: pp. 1117-1122.
Williebeck-Lemair, Marc H.& Zon--Yin Shae, "Videoconferencing Over Packet-Based Networks," IEEE Journal on Selected Areas in Communications, vol. 15, No. 6, Aug. 1997 pp. 1101 to 1114.
Yang, C University of North Texas: "INETPhone: Telephone Services and Servers on the Internet" Apr. 1995; Network Working Group; Request for Comments: 1789; Category: Informational.
Yeager, Nancy, & McGrath, Robert "Web Server Technology, the Advanced Guide for World Wide Web Information Providers" Chapter 6: Searching for Information on the Web, 6.8.2 Using Databases as Indexes to a Document Collection 1996 p. 250.

ART-UNIT: 2663

PRIMARY-EXAMINER: Vu; Huy D.

ABSTRACT:

Telephone calls, data and other multimedia information is routed through a hybrid network which includes transfer of information across the internet. A media order entry captures complete user profile information for a user. This profile information is utilized by the system throughout the media experience for routing, billing, monitoring, reporting and other media control functions. Users can manage more aspects of a network than previously possible, and control network activities from a central site. The hybrid network also contains logic for responding to requests for quality of service and reserving the resources to provide the requested services.

16 Claims, 191 Drawing figures

First Hit Fwd Refs

Generate Collection Print

L11: Entry 3 of 9 File: USPT Oct 8, 2002

DOCUMENT-IDENTIFIER: US 6463504 B1
TITLE: Method and system for dynamically reassigning unused logical volumes from a storage subsystem to an open systems host

Abstract Text (1):
A method and system for reassigning unused logical volumes on a storage subsystem to an open systems host that uses logical unit numbers (LUNs). The storage subsystem and the open systems hosts execute respective software processes, and the open systems host is connected to a storage subsystem via an adapter that is controlled from a support controller. The method and system include sending a communication message from the support controller to the adapter, wherein the communication message reassigns the unused logical volumes to LUNs. The method and system further include updating a logical-to-physical mapping stored within the adapter in response to receiving the communication message, whereby storage capacity of the open systems host is increased without suspending the software processes.

First Hit Fwd Refs

Generate Collection Print

L11: Entry 3 of 9 File: USPT Oct 8, 2002

US-PAT-NO: 6463504
DOCUMENT-IDENTIFIER: US 6463504 B1

TITLE: Method and system for dynamically reassigning unused logical volumes from a storage subsystem to an open systems host

DATE-ISSUED: October 8, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Ishibashi; Karen M.	Boulder	CO		
Gzym; Michael A.	Longmont	CO		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
International Business Machines Corporation	Armonk	NY			02

APPL-NO: 09/ 436072 [PALM]
DATE FILED: November 8, 1999

INT-CL: [07] G06 F 13/00

US-CL-ISSUED: 711/114; 711/112, 711/147, 711/202, 709/205, 709/213, 710/62
US-CL-CURRENT: 711/114; 709/205, 709/213, 710/62, 711/112, 711/147, 711/202

FIELD-OF-SEARCH: 711/112, 711/114, 711/202, 711/147, 711/148, 711/151, 711/163, 711/167, 710/6, 710/62, 710/129, 709/205, 709/213

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4837680	June 1989	Crockett et al.	364/200
☐	5148540	September 1992	Beardsley et al.	395/575
☐	5761448	June 1998	Adamson et al.	395/284
☐	5867686	February 1999	Conner et al.	711/168
☐	6018779	January 2000	Blumenau	710/68
☐	6073218	June 2000	DeKoning et al.	711/150

h e b b g e e e f c e h e ge

☐ 6115772 September 2000 Crater 710/129

OTHER PUBLICATIONS

"Performance Efficient Multiple Logical Unit Number Mapping for Redundant Array of Independent Disks," IBM Technical Disclosure Bulletin vol. 39, No. 05, May 1996.

ART-UNIT: 2187

PRIMARY-EXAMINER: Yoo; Do Hyun

ASSISTANT-EXAMINER: Song; Jasmine

ATTY-AGENT-FIRM: Sawyer Law Group LLP

ABSTRACT:

A method and system for reassigning unused logical volumes on a storage subsystem to an open systems host that uses logical unit numbers (LUNs). The storage subsystem and the open systems hosts execute respective software processes, and the open systems host is connected to a storage subsystem via an adapter that is controlled from a support controller. The method and system include sending a communication message from the support controller to the adapter, wherein the communication message reassigns the unused logical volumes to LUNs. The method and system further include updating a logical-to-physical mapping stored within the adapter in response to receiving the communication message, whereby storage capacity of the open systems host is increased without suspending the software processes.

13 Claims, 6 Drawing figures

First Hit Fwd Refs

Generate Collection Print

L14: Entry 9 of 11 File: USPT Jun 28, 1994

DOCUMENT-IDENTIFIER: US 5325526 A
TITLE: Task scheduling in a multicomputer system

Abstract Text (1):
An improved method of executing a plurality of computer application programs on a multicomputer is disclosed. The present invention pertains to a task scheduling system in a multicomputer having nodes arranged in a network. The present invention comprises an allocator and scheduler component, which comprises processing logic and data for implementing the task scheduler of the present invention. The allocator and scheduler operates in conjunction with a partition to assign tasks to a plurality of nodes. A partition is an object comprising a plurality of items of information and optionally related processing functions for maintaining a logical environment for the execution of tasks of one or more application programs. Application programs are allowed to execute on one or more nodes of a partition. Moreover, a node may be assigned to more than one partition and more than one application program may be loaded on a single node. The allocator and scheduler provides allocator procedures used by application programs for identifying a node or group of nodes for inclusion in a partition. The allocator and scheduler also provides several data areas for the storage of information relevant to the allocation and scheduling of tasks. These data areas of the allocator and scheduler include a partition data area, an application data area, and a layer data area. The present invention provides a means and method for hierarchically linking application programs, layers, and partitions together to provide an optimal execution environment for the execution of a plurality of tasks in a multicomputer.

h e b b cg b cc e

First Hit Fwd Refs

Generate Collection Print

L14: Entry 9 of 11 File: USPT Jun 28, 1994

US-PAT-NO: 5325526
DOCUMENT-IDENTIFIER: US 5325526 A

TITLE: Task scheduling in a multicomputer system

DATE-ISSUED: June 28, 1994

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Cameron; Donald F.	Portland	OR		
Merrow; Thomas E.	Portland	OR		
Pierce; Paul R.	Portland	OR		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Intel Corporation	Santa Clara	CA			02

APPL-NO: 07/ 881879 [PALM]
DATE FILED: May 12, 1992

INT-CL: [05] G06F 9/46

US-CL-ISSUED: 395/650; 364/DIG.1, 364/281.8, 364/281.4, 364/230.3
US-CL-CURRENT: 718/102; 711/173

FIELD-OF-SEARCH: 395/202, 395/650, 395/250, 395/325, 395/575, 371/11, 371/9.1, 371/36

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4658351	April 1987	Teng	395/650
☐	4805107	February 1989	Kieckhafer	395/650

OTHER PUBLICATIONS

Feitelson et al., Distributed Hierarchical Control for Parallel Processing; IEEE Computer, May 1990, pp. 65-76.
Milenkovic, Milan; Operating Systems: Concepts and Design; 1987, pp. 111-117.

h e b b g e e e f c e h e ge

ART-UNIT: 236

PRIMARY-EXAMINER: Heckler; Thomas M.

ATTY-AGENT-FIRM: Blakely, Sokoloff, Taylor & Zafman

ABSTRACT:

An improved method of executing a plurality of computer application programs on a multicomputer is disclosed. The present invention pertains to a task scheduling system in a multicomputer having nodes arranged in a network. The present invention comprises an allocator and scheduler component, which comprises processing logic and data for implementing the task scheduler of the present invention. The allocator and scheduler operates in conjunction with a partition to assign tasks to a plurality of nodes. A partition is an object comprising a plurality of items of information and optionally related processing functions for maintaining a logical environment for the execution of tasks of one or more application programs. Application programs are allowed to execute on one or more nodes of a partition. Moreover, a node may be assigned to more than one partition and more than one application program may be loaded on a single node. The allocator and scheduler provides allocator procedures used by application programs for identifying a node or group of nodes for inclusion in a partition. The allocator and scheduler also provides several data areas for the storage of information relevant to the allocation and scheduling of tasks. These data areas of the allocator and scheduler include a partition data area, an application data area, and a layer data area. The present invention provides a means and method for hierarchically linking application programs, layers, and partitions together to provide an optimal execution environment for the execution of a plurality of tasks in a multicomputer.

22 Claims, 26 Drawing figures

First Hit Fwd Refs

End of Result Set

Generate Collection Print

L1: Entry 1 of 1 File: USPT Sep 12, 2000

DOCUMENT-IDENTIFIER: US 6119131 A
** See image for Certificate of Correction **
TITLE: Persistent volume mount points

Detailed Description Text (30):
The pathname must be parsed to determine which logical volume contains "filename." The parsing of the pathname begins with the operating system replacing the prefix "DosDevices.backslash.C:.backslash." with the device name of the logical volume 205 based on the symbolic link associating DosDevices.backslash.C:.backslash. and the device name. The device name indicates which device driver is then called to continue parsing the pathname. The device driver for logical volume 205 replaces the prefix ".backslash.Programming.backslash." with the device name of the logical volume 206 specified by the .backslash.??.backslash.volume {GUID2} symbolic link, which causes the operating system to invoke the corresponding device driver. In turn, the device driver of the logical volume 206 replaces the prefix ".backslash.Projects.backslash." with the device name of the logical volume 207 specified by the .backslash..backslash.??.backslash.volume {GUID3} symbolic link. The device driver for logical volume 207 recognizes the final portion of the pathname, "filename," as referring to a file on logical volume 207 so the parsing is complete and the device driver for logical volume 207 executes the command against the file.

Detailed Description Text (110):
The mount manager 401 provides a query, GetVolumePathName (directory path), for use by device drivers when parsing pathnames as described above. For a given pathname, GetVolumePathName returns the longest directory prefix in the specified pathname that corresponds to a logical volume. Thus, referring once again to FIG. 2A, GetVolumePathName
(.backslash.DosDevices.backslash.C:.backslash.Programming.backslash.Projec
ts.backslash.filename) returns
"DosDevices.backslash.C.backslash.:.backslash.Programming.backslash.Projec
ts.backslash.," GetVolumePathName
(.backslash.DosDevices.backslash.C:.backslash.Programming.backslash.Projec
ts.backslash.) returns "DosDevices.backslash.C:.backslash.Programming.backslash.,"
and GetVolumePathName
(.backslash.DosDevices.backslash.C:.backslash.Programming.backslash.) returns
".backslash.DosDevices.backslash.C:.backslash.."

h e b b g e e e f c e f b e ge

First Hit Fwd Refs

End of Result Set

Generate Collection Print

L2: Entry 1 of 1 File: USPT Sep 12, 2000

DOCUMENT-IDENTIFIER: US 6119131 A
** See image for Certificate of Correction **
TITLE: Persistent volume mount points

Abstract Text (1):
Information regarding volume mount points hosted by a logical volume are stored on the physical device underlying the logical volume so that the relationships between the host logical volume and target logical volumes mounted on the volume mount points can be reconstituted when the system containing the logical volumes is rebooted, when the underlying physical devices are moved with the system, and when the logical volumes are transported to a different system. A data structure stored on the physical device contains the directory name of the volume mount point and the unique identifier and a globally unique mount name of the target logical volume mounted at the volume mount point. When the target logical volume is present in the system, symbolic links are created to relate the volume mount point name to a device name for the target logical volume so that pathnames containing the directory junction name are resolved correctly. If the target volume is not present in the system, the corresponding symbolic link does not exist, so an incorrect logical volume cannot be mounted onto a volume mount point. Furthermore, because the logical volumes contain the directory junction information, the namespace representing the logical volumes is self-describing so that neither user knowledge nor intervention is required to reconstitute the namespace.

Brief Summary Text (4):
Most operating systems associate a logical unit of mass storage with a device name, such as a physical hierarchical name (.backslash.device.backslash.harddisk0.backslash.partition1.backslash.), and a user-friendly name, such as a drive letter, so that the data on the storage device can be easily accessible by the higher layers of the operating system and user applications. The higher layers of the operating system and user applications assume that the user-friendly names are persistent across boot sessions. In actuality, the names are persistent only as long as the physical configuration of the computer does not change. Persistence cannot be guaranteed because such operating systems assign the user-friendly names in the order in which the storage devices are detected when booting. When the physical locations of the storage devices change, these operating systems will assign the user-friendly names to different devices. Therefore, the consistency of name assignments across multiple boot sessions is not preserved under all circumstances, and the higher operating system layers and user applications will be unable to access the data on the devices without modification or without administrator invention.

Brief Summary Text (5):
The process of associating a logical unit, or volume, with the appropriate underlying physical media is commonly referred to in the art as "mounting" the logical volume on the physical media. The logical volume must be mounted before the data on the physical media can be accessed.

Brief Summary Text (8):

h e b b g e e e f c e f b e ge

Therefore, there is a need in the art for an operating system that provides persistent user-friendly names and guarantees the consistency of volume mount points despite physical configuration changes in the underlying storage media.

Brief Summary Text (12):
computer. Information regarding volume mount points hosted by a logical volume are stored on the physical device underlying the logical volume so that the relationships between the host logical volume and target logical volumes mounted on the volume mount points can be reconstituted when the system containing the logical volumes is rebooted, when the underlying physical devices are moved within the system, and when the logical volumes are transported to a different system. A data structure stored on the physical device contains the directory name of the volume mount point and the unique identifier and a globally unique mount name of the target logical volume mounted at the volume mount point. When the target logical volume is present in the system, symbolic links are created to relate the volume mount point name to a device name for the target logical volume so that pathnames containing the directory junction name are resolved correctly. If the target volume is not present in the system, the corresponding symbolic link does not exist, so an incorrect logical volume cannot be inadvertently mounted onto a volume mount point.

Brief Summary Text (13):
When the physical configuration of the computer changes, the device name changes but the unique volume identifier does not. The unique volume identifier is used to locate the appropriate mount name and/or drive letter in its data structure and a new symbolic link is created with the new device name so that the symbolic link resolves the mount name and/or drive letter to the correct logical volume under all circumstances.

Brief Summary Text (14):
Because the logical volume is created through its unique volume identifier and is not reliant on the devices being located in any particular order in the system, or being discovered in any particular order during the boot process, or being present only during the boot process, changes in the physical configuration of the computer between boots, or during a boot session, have no effect on the higher layers of the operating system and user applications which rely on the redirected name. Thus, the level of indirection provided by the persistent mount names guarantees that the higher layers of the operating system and user applications will be able to access data on a logical volume for the life of the logical volume without modifications. Furthermore, because the logical volumes contain the volume mount point information, neither user knowledge nor intervention is required to reconstitute the namespace representing the logical volumes.

Detailed Description Text (16):
Physical media in a computer contains one or more logical volumes. The process of associating a logical volume with the appropriate underlying physical media is commonly referred to in the art as "mounting" the logical volume on the physical media. The logical volume must be mounted before the data on the physical media can be accessed.

Detailed Description Text (18):
A logical volume represents portions of one or more physical devices which holds the data for the files in the file system. The hierarchy that encompasses all logical volumes in a computer system is the "global" namespace of the file system. Each logical volume in the computer also has its own "local" namespace which defines the directories and files present in the logical volume. A volume mount point in the local namespace of one logical volume is used to "graft" the local namespace of a second logical volume into the namespace of the first logical volume. Thus, a volume mount point is a portal or entry from the first logical volume into the data in the second logical volume. Volume mount points permit a

user to be presented with a single view of multiple logical volumes so that the user does not have to specify the path to each individual logical volume in order to access data on the logical volumes. The user's view is independent of the number of the logical volumes and the arrangement of the physical devices underlying the logical volumes.

Detailed Description Text (20):
In the present invention, unique volume identifiers are used to identify and track logical volumes created from storage devices in a computer and to associate persistent "redirected" mount names with a logical volume to create a self-describing namespace which presents a consistent view of the logical volumes in the computer despite physical configuration changes. In the embodiments described herein, the unique volume identifiers and "redirected" mount names are stored in persistent data structure that is created and maintained by a mount manager. The mount manager is shown and described for the sake of clarity in understand the invention but the invention can be practiced in any environment which provides the functions necessary to manage the unique volume identifiers, the redirected mount names, and the associations with the logical volumes.

Detailed Description Text (21):
FIG. 2A shows one embodiment of a logical volume mounting subsystem 250 in a computer 200 such as local computer 20 or remote computer 49 in FIG. 1. The physical media, such as hard disk drive 27 in local computer, contains one or more logical volumes. The logical volume mounting subsystem 250 comprises a mount manager 201 and a persistent mount manager data structure 203, and is responsible for associating user-friendly "redirected" names 213, used by the higher layers of the operating system

Detailed Description Text (22):
and user applications, with mounted logical storage volumes 205, 206 207, and 208 so that the data on the underlying physical devices can be accessed through the redirected names. In FIG. 2A, the redirected names 233 are shown as drive letters, such as commonly used by personal computer applications, and mount names which have a format similar to a drive letter. The redirected names are not limited to the address format shown in FIG. 2A but are adaptable to the various formats of user-friendly names prevalent in all operating systems, as will be readily apparent to one skilled in the art. The logical storage volumes 205-208 combined make up the global namespace 260 of the computer 200.

Detailed Description Text (23):
The computer operating system (not shown) creates the logical volumes 205-208 from physical removable or fixed media devices, such the hard disk drive 27. Each logical volume is identified by a unique volume identifier 223 which is stored on the physical device, or devices, that make up the logical volume and which is guaranteed to be unique on the particular computer. The unique volume identifier for each logical volume 205, 216, 217 and 208 is shown stored in data structures 215, 216, 217 and 218, respectively, in FIG. 2A. Also stored in the data structures is a mount name for the volume as described further below.

Detailed Description Text (28):
Because the unique volume identifier 223 for a logical volume is associated with a redirected name 213 in the persistent mount manager data structure 203, the logical volume will be always be assigned the same redirected name, either a drive letter or a mount name, even if the logical volume is presented to the mount manager 201 in a different order. The order in which the logical volumes are presented is dependent upon the order in which they are detected by the operating system so the order changes if the underlying physical devices are rearranged in the computer between boots. The device name of the logical volumes also change when the physical configuration changes. Because the logical volume is identified by the unique volume identifiers 213 rather than the device names, consistency between the mount

names and/or drive letters and the logical volumes is assured across boot sessions regardless of the underlying physical configuration of the computer or the order in which the devices are recognized as long as the logical volume is valid.

Detailed Description Text (41):
The system level overview of the operation of an exemplary embodiment of the invention has been described in this section of the detailed description. A mount manager and supporting data structures enable consistent identification and addressing of logical volumes despite physical configuration changes for the life of the logical volumes through the use of persistent user-friendly redirected names. The mount manager also stores directory junction information on logical volumes so that the namespace represented by the logical volumes is self-describing. While the invention is not limited to any particular arrangement of data in the data structures or any particular user-friendly format for the redirected names, for sake of clarity exemplary embodiments of persistent and in-memory data structures and redirected names have been illustrated and described.

Detailed Description Text (46):
If the entry contains a redirected name(s), the mount manager 201 updates the in-memory data structure with the device name at step 315, and causes the operating system to create a symbolic link between the redirected name(s) and the boot session device name (step 317). Thus, the redirected name(s) used by the higher layers of the operating system and user applications are preserved across boot sessions, even though the device names may change when the physical configuration of the computer is modified. In an alternate embodiment in which the in-memory data structure does not contain the device name field, the mount manager 201 skips step 315, as both the in-memory and persistent data structures already contain the correct unique volume identifier and redirected name(s) (illustrated by a phantom logic flow path in FIG. 3A).

Detailed Description Text (56):
In this section of the detailed description, a particular implementation of the invention is described that executes as part of the Microsoft Windows NT 5.0 operating system kernel. In the implementation illustrated in FIG. 4, the mount manager 401 and four other kernel modules work together to provide a user with access to data stored on a physical storage device 411 (shown as a fixed hard disk): a plug and play manager 403, an object manager 405, a partition manager 407, and at least one volume manager 409.

Detailed Description Text (60):
Because NT 5.0 is an object-based operating system, every device, either physical, logical, or virtual, within the system is represented by a device object. The objects are organized into a device hierarchy in a global namespace controlled by the object manager 405. The object manager 405 is also responsible for creating and maintaining symbolic link objects which serve as aliases for named device objects. The mount manager redirected name is represented in the namespace by a symbolic link object which contains the non-persistent device name of the corresponding logical volume. Thus, an "Open" command operating on a redirected name symbolic link object is the same as an "Open" command on the logical volume device object having the device name contained in the symbolic link object.

Detailed Description Text (61):
The partition manager 407 is responsible for handling device objects associated with logical divisions, partitions 412, 413, 414, and 415, of a physical device 411. The partitions 412-415 are created when the physical device 411 is formatted. The partition 412 comprises the entire physical device 411, while the partitions 413-415 are sub-divisions of the physical device 411. A device driver (not shown) for the physical device 411 "enumerates" corresponding partition device objects 421, 422, 423, and 424 when the computer is booted. The partition manager 407 and at least one volume manager 409 cooperate to create logical volumes from the

partitions 413-415. The composition of a logical volume is defined when the physical device is formatted, and can comprise one or more partitions. Additionally, one partition can comprise more than one logical volume. A unique volume identifier for the logical volume is stored in a privileged section of the physical device, or devices, that contain the partitions making up the logical volume. The volume manager, or the device driver in the case of a removable device, responsible for the volume device object creates the unique volume identifier.

Detailed Description Text (63):
When the physical device 411 is detected by the plug and play manager 403 upon booting the system, the plug and play manager 403 determines the formatted characteristics of the physical device 411. The plug and play manager 403 loads the appropriate device driver to handle I/O access to the device. The device driver enumerates the partition device objects 421-424 used to access the data. As each partition device object 422-424 not representative of the entire device is enumerated by the device driver, the partition manager 407 "captures" the partition device object 422-424 before the driver registers the object with the plug and play manager 403. The partition manager 407 presents each partition device object 422-424 to the volume managers 409 in the order in which the volume managers 409 arrived in the system. Because each partition device object 424-424 is associated with at least one logical volume, the volume manager 409 responsible for the corresponding logical volume(s) accepts the device object.

Detailed Description Text (83):
QueryDeviceName and QueryUniqueId return the device name and unique volume identifier for the specified volume device object. QueryDesiredName returns a recommended redirected name(s) for the mount manager 401 to use in the event that the mount manager data structure(s) 441 does not yet contain any entries for the specified volume device object. A physical device that supports QueryDesiredName usually stores the desired name(s) in the same privileged area of the physical device as the unique volume identifier.

Detailed Description Text (85):
LinkCreated and LinkDeleted are used by the mount manager 401 to inform the volume device object of the creation and deletion of the redirected names and the symbolic link objects that reference the volume device object. A physical device that supports LinkCreated and LinkDeleted can use the information to update any redirected name(s) it has internally stored.

Detailed Description Text (109):
SetVolumeMountPoint causes the mount manager 401 to update the data structure 441 with the specified mount name and directory name, if the directory is empty, or specified mount name and drive letter if the drive letter is not in use. The mount manager 401 requests that the object manager 405 create the appropriate symbolic link object to represent the relationship between the device name of the logical volume identified by the mount name or drive letter. The unique identifier and the mount name are also stored on the physical device underlying the host logical volume in a named datastream ".backslash.:$MountMgrRemoteDatabase." DeleteVolumeMountPoint causes the mount manager 401 to delete the appropriate entries from the data structure 441 and, if the input argument is a directory path, to delete the directory junction data from the named datastream. The mount manager 401 also requests that the object manager 405 destroy the corresponding symbolic link object.

Detailed Description Text (111):
Thus, the invention guarantees the same redirected name(s) will be associated with the same logical volume across any number of boots or reconfigurations as long as the logical volume itself remains valid. The persistence of the redirected names enables the creation of a self-describing namespace in a computer so that the underlying physical devices can be moved within a computer and between computers

without the necessity of operator intervention. Therefore, the invention guarantees that I/O commands on a redirected name are resolved through the symbolic link objects to the current device name for the correct logical volume without modification to the higher layers of the operating system and user applications despite changes in the physical configuration of the computer.

Detailed Description Text (113):
Unique volume identifiers for logical volumes are have been described as used to associate logical volumes created from computer system storage devices with persistent "redirected" mount names to create a self-describing namespace on the computer. Information regarding volume mount points hosted by a logical volume are stored on the physical device underlying the logical volume so that the relationships between the host logical volume and target logical volumes mounted on the volume mount points can be reconstituted when the system containing the logical volumes is rebooted, when the underlying physical devices are moved with the system, and when the logical volumes are transported to a different system.

Detailed Description Text (114):
A data structure stored on the physical device contains the directory name of the volume mount point and the unique identifier and a globally unique mount name of the target logical volume mounted at the volume mount point. When the target logical volume is present in the system, symbolic links are created to relate the volume mount point name to a device name for the target logical volume so that pathnames containing the directory junction name are resolved correctly. If the target volume is not present in the system, the corresponding symbolic link does not exist, so an incorrect logical volume cannot be mounted onto a volume mount point.

Detailed Description Text (115):
When the physical configuration of the computer changes, the device name changes but the unique volume identifier does not. The unique volume identifier is used to locate the appropriate mount name and/or drive letter in its data structure and a new symbolic link is created with the new device name so that the symbolic link resolves the mount name and/or drive letter to the correct logical volume under all circumstances.

CLAIMS:

1. A computerized method for managing a self-describing namespace comprising the steps of:

locating a volume mount point in the namespace;

creating, in a host logical volume for the volume mount point, a persistent association between the volume mount point and a unique mount name for a target logical volume mounted at the volume mount point if there is no existing association, wherein the unique mount name is independent of the logical volume's physical location in a computer; and

linking the unique mount name to a non-persistent device name for the target logical volume so that a reference to the unique mount name accesses data on the logical volume.

4. The computerized method of claim 1, wherein the step of creating the persistent association comprises the step of:

storing, on a physical device corresponding to the host logical volume, a directory junction data structure comprising the mount name for the logical volume.

10. The computer system of claim 8, wherein the persistent association is stored as

a data structure on a physical device corresponding to the host logical volume.

11. A computer-readable medium having computer-executable components comprising:

a plug and play manager for detecting the presence of a physical device in a computer system and for assigning a device driver responsibility for controlling access to the physical device;

a partition manager communicatively coupled to the device driver for capturing partition device objects enumerated from the physical device by the device driver, wherein each partition device object corresponds to a portion of the physical device, the partition manager further communicatively coupled to the plug and play manager;

a volume manager communicatively coupled to the partition manager for creating a volume device object from at least one partition device object captured by the partition manager, for assigning a device name to the logical volume represented by the volume device object, and further communicatively coupled to the plug and play manager for registering the creation of the volume device object, wherein the volume device object comprises the device name and a unique volume identifier for the logical volume;

a mount manager communicatively coupled to the plug and play manager for receiving notification of the creation of the volume device object, for establishing a persistent association between the unique volume identifier of the volume device object and a unique mount name, for establishing a persistent association between the unique volume identifier and a drive letter when requested by the plug and play manager, and for establishing a persistent association on a host logical volume between a volume mount point on the host logical volume and a unique mount name for a logical volume object for a target logical volume mounted at the volume mount

point; and

an object manager communicatively coupled to the partition manager, the volume manager, and the mount manager for managing the partition device objects and the volume device object, for creating a symbolic link object for the mount name that causes a reference to the mount name to be redirected to the volume device object, for creating a symbolic link object for the drive letter that causes a reference to the drive letter to be redirected to the volume device object, and for creating a symbolic link object for the mount name that causes a reference to the mount name to be redirected to the volume device object for the target logical volume.

device. The remote computer 49 may be another computer, a server, a router, a network PC, a client, a peer device or other common network node, and typically includes many or all of the elements described above relative to the computer 20, although only a memory storage device 50 has been illustrated in FIG. 1. The logical connections depicted in FIG. 1 include a local-area network (LAN) 51 and a wide-area network (WAN) 52. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.

Detailed Description Text (12):
When used in a LAN-networking environment, the computer 20 is connected to the local network 51 through a network interface or adapter 53, which is one type of communications device. When used in a WAN-networking environment, the computer 20 typically includes a modem 54, a type of communications device, or any other type of communications device for establishing communications over the wide area network 52, such as the Internet. The modem 54, which may be internal or external, is connected to the system bus 23 via the serial port interface 46. In a networked environment, program modules depicted relative to the personal computer 20, or portions thereof, may be stored in the remote memory storage device. It is appreciated that the network connections shown are exemplary and other means of and communications devices for establishing a communications link between the computers may be used.

Detailed Description Text (13):
The hardware and operating environment in conjunction with which embodiments of the invention may be practiced has been described. The computer in conjunction with which embodiments of the invention may be practiced may be a conventional computer, a distributed computer, or any other type of computer; the invention is not so limited. Such a computer typically includes one or more processing units as its processor, and a computer-readable medium such as a memory. The computer may also include a communications device such as a network adapter or a modem, so that it is able to communicatively couple to other computers.

CLAIMS:

8. A computer system comprising:

a processing unit;

a system memory coupled to the processing unit through a system bus;

a computer-readable medium coupled to the processing unit through a system bus; and

a mount manager executed from the computer-readable medium by the processing unit, wherein the mount manager causes the processing unit to create a persistent association on a host logical volume between a volume mount point on the host logical volume and a unique mount name for a target logical volume mounted at the volume mount point if there is no existing association, and establish a symbolic link between the mount name and a non-persistent device name for the target logical volume upon each boot of the computer so that a reference to the mount name is correctly resolved by the processing unit to the target logical volume.

11. A computer-readable medium having computer-executable components comprising:

a plug and play manager for detecting the presence of a physical device in a computer system and for assigning a device driver responsibility for controlling access to the physical device;

a partition manager communicatively coupled to the device driver for capturing partition device objects enumerated from the physical device by the device driver,

wherein each partition device object corresponds to a portion of the physical device, the partition manager further communicatively coupled to the plug and play manager;

a volume manager communicatively coupled to the partition manager for creating a volume device object from at least one partition device object captured by the partition manager, for assigning a device name to the logical volume represented by the volume device object, and further communicatively coupled to the plug and play manager for registering the creation of the volume device object, wherein the volume device object comprises the device name and a unique volume identifier for the logical volume;

a mount manager communicatively coupled to the plug and play manager for receiving notification of the creation of the volume device object, for establishing a persistent association between the unique volume identifier of the volume device object and a unique mount name, for establishing a persistent association between the unique volume identifier and a drive letter when requested by the plug and play manager, and for establishing a persistent association on a host logical volume between a volume mount point on the host logical volume and a unique mount name for a logical volume object for a target logical volume mounted at the volume mount

point; and

an object manager communicatively coupled to the partition manager, the volume manager, and the mount manager for managing the partition device objects and the volume device object, for creating a symbolic link object for the mount name that causes a reference to the mount name to be redirected to the volume device object, for creating a symbolic link object for the drive letter that causes a reference to the drive letter to be redirected to the volume device object, and for creating a symbolic link object for the mount name that causes a reference to the mount name to be redirected to the volume device object for the target logical volume.

First Hit Fwd Refs

End of Result Set

Generate Collection Print

L1: Entry 1 of 1 File: USPT Sep 12, 2000

DOCUMENT-IDENTIFIER: US 6119131 A

** See image for Certificate of Correction **

TITLE: Persistent volume mount points

Detailed Description Text (7):

The exemplary hardware and operating environment of FIG. 1 for implementing the invention includes a general purpose computing device in the form of a computer 20, including a processing unit 21, a system memory 22, and a system bus 23 that operatively couples various system components, including the system memory 22, to the processing unit 21. There may be only one or there may be more than one processing unit 21, such that the processor of computer 20 comprises a single central-processing unit (CPU), or a plurality of processing units, commonly referred to as a parallel processing environment. The computer 20 may be a conventional computer, a distributed computer, or any other type of computer; the invention is not so limited.

Detailed Description Text (9):

The hard disk drive 27, magnetic disk drive 28, and optical disk drive 30 are connected to the system bus 23 by a hard disk drive interface 32, a magnetic disk drive interface 33, and an optical disk drive interface 34, respectively. The drives and their associated computer-readable media provide nonvolatile storage of computer-readable instructions, data structures, program modules and other data for the computer 20. It should be appreciated by those skilled in the art that any type of computer-readable media which can store data that is accessible by a computer, such as magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, random access memories (RAMs), read only memories (ROMs), and the like, may be used in the exemplary operating environment.

Detailed Description Text (10):

A number of program modules may be stored on the hard disk, magnetic disk 29, optical disk 31, ROM 24, or RAM 25, including an operating system 35, one or more application programs 36, other program modules 37, and program data 38. A user may enter commands and information into the personal computer 20 through input devices such as a keyboard 40 and pointing device 42. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 21 through a serial port interface 46 that is coupled to the system bus, but may be connected by other interfaces, such as a parallel port, game port, or a universal serial bus (USB). A monitor 47 or other type of display device is also connected to the system bus 23 via an interface, such as a video adapter 48. In addition to the monitor, computers typically include other peripheral output devices (not shown), such as speakers and printers.

Detailed Description Text (11):

The computer 20 may operate in a networked environment using logical connections to one or more remote computers, such as remote computer 49. These logical connections are achieved by a communication device coupled to or a part of the computer 20, the local computer; the invention is not limited to a particular type of communications

L9: Entry 9 of 62 File: USPT Nov 25, 2003

DOCUMENT-IDENTIFIER: US 6654881 B2
TITLE: Logical volume mount manager

Brief Summary Text (4):
Most operating systems identify a logical unit of mass storage through a "well-known" and system compatible name which defines an actual physical path to the logical unit, i.e., .backslash.device0.backslash.partition1.backslash.. The operating system then associates a user-friendly name, such as a drive letter, with the well-known name so that the data on the storage device can be easily accessible by higher layers of the operating system and user applications. The higher layers of the operating system and applications assume that the well-known names, and thus the associated user-friendly names, are persistent across boot sessions. In actuality, the names are persistent only as long as the physical configuration of the computer does not change. Persistence cannot be guaranteed because such operating systems assign the well-known names in the order in which the storage devices are detected when booting. When the physical locations of the storage devices change, these operating systems will assign the well-known names to different devices. Therefore, the consistency of name assignments across multiple boot sessions is not preserved under all circumstances, and the higher operating system layers and user applications will be unable to access the data on the devices without modification.

Brief Summary Text (9):
A logical volume mount manager is responsible for identifying and tracking logical volumes created from a physical storage device by the operating system, and for determining a redirected name for a logical volume which is used by higher layers of the operating system and user applications. The mount manager builds and maintains a persistent data structure based on a unique volume identifier which identifies the logical volume. Optionally, the mount manager also creates an in-memory data structure as well. Each entry in the data structure(s) consists of the redirected name and the unique volume identifier for a logical volume so that the redirected name persists across boot sessions. Because the operating system addresses a logical volume through a non-persistent device name, the mount manager causes the operating system to create a symbolic link between the device name and the redirected name when the mount manager first identifies the logical volume during a boot session so that the higher layers of the operating system and user applications can access the logical volume through the persistent redirected name.

Detailed Description Text (15):
A system level overview of the operation of an exemplary embodiment of the invention is described by reference to FIGS. 2A-C. FIG. 2A shows one embodiment of a logical volume mounting subsystem 200 executing in a computer such as local computer 20 or remote computer 49 in FIG. 1. The physical media, such as hard disk drive 27 in local computer 20, contains one or more logical volumes. The process of associating a logical volume with the appropriate underlying physical media is commonly referred to in the art as "mounting" the logical volume on the physical media. The logical volume must be mounted before the data on the physical media can be accessed.

Detailed Description Text (16):

The logical volume mounting subsystem 200 shown in FIG. 2A comprises a mount manager 201 and a persistent mount manager data structure 203, and is responsible for associating "redirected" names, used by the higher layers of the operating system and user applications, with mounted logical storage volumes 207, 208 and 209 so that the data on the underlying physical devices can be accessed through the redirected names. In FIG. 2A, the redirected names are represented by drive letters, such as commonly used by personal computer applications. The redirected names are not limited to drive letters as will be readily apparent to one skilled in the art and described in more detail below in conjunction with FIGS. 2B and 2C.

Detailed Description Text (17):
The operating system 205 creates the logical volumes 207-209 from removable or fixed physical media devices, such as hard disk drive 27. Each logical volume is identified by a unique volume identifier, such as 994 for logical volume 207, which is stored on the physical device, or devices, that make up the logical volume, and which is guaranteed to be unique on the particular computer. Each logical volume is also assigned a device name, such as Vol1 for logical volume 207, during the boot process. The device name can change across boot sessions but is unique for a particular boot session.

Detailed Description Text (44):
In this section of the detailed description, a particular implementation of the invention is described that executes as part of the Microsoft Windows NT 5.0 operating system kernel. In the implementation illustrated in FIG. 4, the mount manager 401 and four other kernel modules work together to provide a user with access to data stored on a physical storage device 411 (shown as a fixed hard disk): a plug and play manager 403, an object manager 405, a partition manager 407, and at least one volume manager 409. The mount manager 401 is not limited to use with only devices that adhere to the partition manager and volume manager architectures described below. The mount manager 401 will manage any device which registers with the plug and play manager 403 which has some mechanism for reporting a device name and a unique identifier that is persistent between boots. The partition manager 407 and the volume manager 409 are shown and described for the sake of clarity in understanding the invention.

Detailed Description Text (48):
When the partition manager 407 is initialized, it requests notification from the plug and play manager of all volume managers 409 registered in the system. As each volume manager 409 registers, the plug and play system notifies the partition manager 407 which maintains a list of the volume managers 409 ordered by their arrival in the system.

Detailed Description Text (49):
When the physical device 411 is detected by the plug and play manager 403 upon booting the system, the plug and play manager 403 determines the formatted characteristics of the physical device 411. The plug and play manager 403 loads the appropriate device driver to handle I/O access to the device. The device driver enumerates the partition device objects 421-424 used to access the data. As each partition device object 422-424 not representative of the entire device is enumerated by the device driver, the partition manager 407 "captures" the partition device object 422-424 before the driver registers the object with the plug and play manager 403. The partition manager 407 presents each partition device object 422-424 to the volume managers 409 in the order in which the volume managers 409 arrived in the system. Because each partition device object 424-424 is associated with at least one logical volume, the volume manager 409 responsible for the corresponding logical volume(s) accepts the device object.

Detailed Description Text (50):
When a volume manager 409 has received a sufficient number of partition device objects corresponding to a particular logical volume, the volume manager 409

assigns a device name to the logical volume and enumerates a volume device object 431-432 for the logical volume containing the device name and the unique volume identifier for the logical volume. In the NT 5.0 embodiment, the device name is guaranteed to be unique only during a boot session, while the unique volume identifier is guaranteed to be unique across boot sessions. A counted string is used as the unique volume identifier in the NT 5.0 environment but a fixed length string can be equally applicable in other operating system environments. The counted string is as long as necessary to uniquely identify the device in the computer across multiple boot sessions. The volume device object 431-432 is stored by the object manager 405 in the device hierarchy by its device name. The volume manager 409 informs the plug and play manager 403 of the creation of the volume device object 431-432.

Detailed Description Text (51):
Each volume device object 431-432 is presented to the mount manager 401 by the plug and play manager 403. The mount manager 401 queries the volume device object 431-432 for its device name and unique volume identifier. Because of the indeterminate length of the unique volume identifier, the volume device object returns a byte count along with the string.

Detailed Description Text (54):
In order to facilitate the assignment of user-friendly drive letters to logical volumes, the mount manager data structure(s) 441 contains an entry for each of the twenty-four drive letters assignable to fixed hard disks, i.e., .backslash.DosDevices.backslash.C: .backslash.-.backslash.DosDevices.backslash.Z:.backslash.. The entries are sorted in alphabetical order. Upon the initial boot of the computer, only the logical boot volume is assigned a drive letter (such as .backslash.DosDevices.backslash.C:.backslash.). When a drive letter is requested for a logical volume by the plug and play manager 403 during the initial boot process, the mount manager 401 assigns the next available drive letter by storing the unique volume identifier in the corresponding entry in the data structure(s) 441. The mount manager 401 requests that the object manager 405 create a symbolic link object representing the association between the drive letter and the volume device name.

Detailed Description Text (55):
On subsequent boots, each logical device is assigned its previous drive letter if one is present in the data structure(s) 441. If a new logical device is introduced into the system during the boot process, the plug and play manager 403 must request the assignment of a drive letter. On the other hand, a new logical volume that is introduced during a boot session is automatically assigned the next available drive letter.

Detailed Description Text (56):
The plug and play manager 403 also informs the mount manager 401 when a logical volume will be temporarily removed from the boot session. The mount manager 401 deletes the device names, if present, from the data structure(s) 441. The mount manager 401 also causes the object manager 405 to retire the symbolic link objects relating the volume device name and the drive letter and/or persistent mount name.

Detailed Description Text (61):
where the pointer to a device object is passed to the mount manager 401 by the plug and play manager 403.

Detailed Description Text (66):
The mount manager 401 also provides an API with the plug and play manager 403 for managing the association between unique volume identifiers and the redirected names: QueryPoints (drive letter/*, unique volume identifier/*, device name/*); DeletePoints (drive letter/*, unique volume identifier/*, device name/*);

DeletePointsDBOnly (drive letter/*, unique volume identifier/*, device name/*); CreatePoint (drive letter, device name); NextDriveLetter(device name); AutoDriveLetter; FindFirstVolume; and FindNextVolume (persistent mount name).

Detailed Description Text (67):
QueryPoints is called by the plug and play manager 403 to retrieve the entry in the mount manager data structure 441 for a logical volume. The plug and play manager 403 specifies the drive letter, the unique volume identifier, or any device name associated with the logical volume as search criteria to be used by the mount manager 401(* is a "wild card" that matches all entries).

Detailed Description Text (70):
In order to preserve the historical drive letter assignments across boot sessions, the mount manager 401 does not automatically assign a drive letter to a logical volume during the boot process unless the logical volume had previously been assigned a drive letter. Therefore, the plug and play manager 403 uses NextDriveLetter to request that the mount manager 401 assign a drive letter to the logical volume associated with the device name specified in the call. NextDriveLetter returns the current drive letter and an indication of whether a drive letter was assigned. A drive letter cannot be assigned if no drive letter is available or if the logical volume represented by the device name is already assigned a drive letter. The plug and play manager 403 can also use AutoDriveLetter once the historical assignments have been made to request the mount manager 401 assign drive letters to all subsequent logical volumes upon arrival.

CLAIMS:

28. A computer-readable medium having computer-executable components comprising: a plug and play manager for detecting the presence of a physical device in a computer system and for assigning a device driver responsibility for controlling access to the physical device; a partition manager communicatively coupled to the device driver for capturing partition device objects enumerated from the physical device by the device driver, wherein each partition device object corresponds to a portion of the physical device, the partition manager further communicatively coupled to the plug and play manager; a volume manager communicatively coupled to the partition manager for creating a volume device object from at least one partition device object captured by the partition manager, for assigning a non-persistent device name to the logical volume represented by the volume device object, and further communicatively coupled to the plug and play manager for registering the creation of the volume device object, wherein the volume device object comprises the non-persistent device name and a unique volume identifier for the logical volume; a mount manager communicatively coupled to the plug and play manager for receiving notification of the creation of the volume device object, and for establishing a persistent association between the unique volume identifier of the volume device object and a unique mount manager identifier, wherein the unique mount manager identifier is persistent and unique across all computers; and an object manager communicatively coupled to the partition manager, the volume manager, and the mount manager for managing the partition device objects and the volume device object, and for creating a symbolic link object for the mount manager identifier that causes a reference to the mount manager identifier to be redirected to the volume device object.

34. The computer-readable medium of claim 28, wherein the mount manager returns the unique volume identifier, the mount manager identifier, and the device name for the logical volume when requested by the plug and play manager.

First Hit Fwd Refs

Generate Collection Print

L9: Entry 9 of 62 File: USPT Nov 25, 2003

US-PAT-NO: 6654881
DOCUMENT-IDENTIFIER: US 6654881 B2

TITLE: Logical volume mount manager

DATE-ISSUED: November 25, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Cabrera; Luis Felipe	Bellevue	WA		
Kusters; Norbert P.	Woodinville	WA		
Wieland; Peter W.	Bellevue	WA		
Rinne; Robert D.	Seattle	WA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Microsoft Corporation	Redmond	WA			02

APPL-NO: 09/ 096772 [PALM]
DATE FILED: June 12, 1998

PARENT-CASE:
RELATED APPLICATIONS This application is related to the co-assigned and co-filed U.S. patent applications titled "Persistent Names for Logical Volumes" (U.S. patent application Ser. No. 09/096,540, now U.S. Pat. No. 6,496,839) and "Persistent Volume Mount Points" (U.S. Pat. No. 6,119,131), which are hereby incorporated by reference.

INT-CL: [07] G06 F 9/00

US-CL-ISSUED: 713/100
US-CL-CURRENT: 713/100

FIELD-OF-SEARCH: 709/321, 709/327, 710/10, 713/1, 713/2, 713/100, 711/202, 711/4, 711/5, 707/1, 707/200

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	5032979	July 1991	Hecht et al.	364/200

h e b b g e e e f c e e e c e ge

☐	5465365	November 1995	Winterbottom	395/600
☐	5511227	April 1996	Jones	710/9
☐	5581760	December 1996	Atkinson et al.	395/700
☐	5623666	April 1997	Pike et al.	395/616
☐	5671414	September 1997	Nicolet	395/684
☐	5675795	October 1997	Rawson, III et al.	395/652
☐	5689706	November 1997	Rao et al.	395/617
☐	5692128	November 1997	Bolles et al.	395/200.09
☐	5724512	March 1998	Winterbottom	395/200.12
☐	5778385	July 1998	Pratt	707/200
☐	5870734	February 1999	Kao	707/2
☐	5881285	March 1999	DeLeeuw	709/321
☐	5896546	April 1999	Monahan et al.	710/10
☐	5931935	August 1999	Cabrera et al.	710/260
☐	5991777	November 1999	Momoh et al.	707/205
☐	6026402	February 2000	Vossen et al.	707/9
☐	2002/0078335	June 2002	Cabrera et al.	713/1

ART-UNIT: 2151

PRIMARY-EXAMINER: Lao; Sue

ATTY-AGENT-FIRM: Merchant & Gould P.C.

ABSTRACT:

A mount manager and supporting data structures enable automatic identification and re-establishment of logical volumes on non-removable storage devices in a computer system across multiple reboots and reconfigurations. The mount manager generates a redirected name for a new logical volume when a unique volume identifier is presented to the mount manager by the operating system. The mount manager stores the unique volume identifier and the associated redirected name in a persistent mount manager data structure The mount manager establishes a symbolic link between the persistent redirected name, which is used by higher layers of the operating system and user applications to address the logical volume, and a non-persistent device name used by the operating system. During the boot process, the mount manager uses the data structure entries identified by the unique volume identifiers of the arriving logical volumes to reconstruct the symbolic links so that references to the redirected name will resolve to the correct non-persistent device name. When the system undergoes physical reconfiguration, the mount manager associates an existing redirected name to a different non-persistent device name if the unique volume identifier is present in the data structure. In this fashion, logical volumes can be removed and restored in the computer without the knowledge of higher layers of the operating system and user applications. Optionally, the mount manager builds an in-memory data structure from the persistent data structure to increase the speed of the identification process.

h e b b g e e e f c e e e c e ge